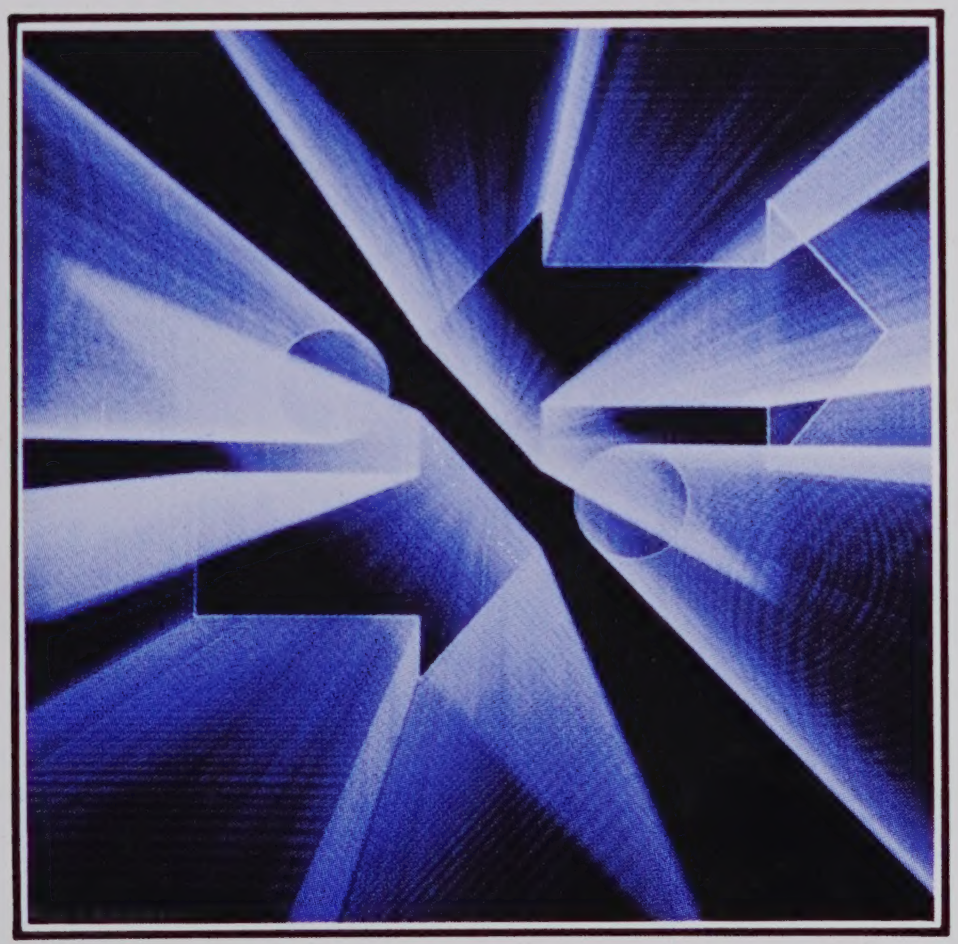
Multiprogrammer



Documentation & Software

HEWLETT PACKARD

Digitized by the Internet Archive
in 2025

https://archive.org/details/manualsplus_05517

HEWLETT PACKARD

14752A
Programming Reference Manual

Copyright © Hewlett-Packard Company 1984

Part No. 14752-90001
Microfiche No. 14752-90002

Printed: November, 1985

Printing History

Each new edition of this manual will incorporate all changes since the previous edition. Update packets or change sheets may be issued between editions to correct or add information. The manual printing data indicates the current edition. The printing date changes with each new edition or update. (Minor corrections incorporated at reprint do not cause a new edition.)

December 1984... First Edition
November 1985...Update 1.....Programming Package Disc Rev. A.01
Utility Programs Disc Rev. A.01

Software Revisions

Revision codes on each 14752A CAT Programming Package disc indicate changes to the software. Each new revision of the software will incorporate all changes since the previous revision. (Minor corrections incorporated at any time do not cause a new revision). Be sure to include the disc's revision code and the date of purchase on the Software Registration Card returned to Hewlett-Packard.

NOTICE

The information contained in this document is subject to change without notice.

CONTENTS

Chapter 1
Introduction

Computer Aided Test....1-1
The Multiprogrammer Computer Aided Test System....1-1
The 14752A CAT Programming Package....1-3
The MENUER Subprogram....1-3
Compiled PASCAL Subprogram I/O Library....1-4
The Utility Programs....1-5
6944S/14752A Specifications....1-6
14752A Documentation....1-9

Chapter 2
Assembling the System

Introduction....2-1
Initial Inspection....2-1
Setting Up the System Hardware....2-4
CAT System Computer Installation....2-4
CAT System Accessory Installation....2-4
Model 216 Computer Installation....2-5
Model 217 Computer Installation....2-6
Model 220 Computer Installation....2-6
Model 226 and 236 Computer Installation....2-7
Model 237 Computer Installation....2-7
9888A Bus Expander....2-8
Model 310 Computer Installation....2-8
Model 320 Computer Installation....2-9
HP 98568A I/O Expander....2-10
Installing Memory Cards....2-11
Installing the Dedicated 98624A HP-IB Interface Card....2-13
Installing the 98633A Multiprogrammer Interface Card....2-14
CAT System Peripheral Installation....2-15
System Printer....2-15
Flexible Disc Drive....2-15
3478A Digital Multimeter....2-15
HP-IB Interface Restrictions....2-16
6942A/6943A Multiprogrammer Installation....2-17
6942A Multiprogrammer System Connections....2-21
6944A Multiprogrammer Installation....2-24
6944S Multiprogrammer System Connections....2-27
Setting Up the System Software....2-29

Chapter 3
Getting Started

Introduction....3-1

The 14752A Software....3-2
The 14752A CAT Programming Package Disc....3-2
The 14752A Utility Programs Disc....3-3
The EXAMPLE Application Program....3-4
Loading the EXAMPLE Program....3-4
Editing the EXAMPLE Program....3-4
The COM Statement....3-9
The Test_multi Statement....3-10
The Configure Statement....3-11
The System_init Statement....3-12
The User's Application Program....3-13
The Error and Interrupt Handlers....3-13
Storing the Application Program....3-13
Loading the 14752A Library and Running Your Program....3-14
Configuration Phase....3-16
Menu Sequence....3-16
The Menu Screens....3-17
The Welcome Menu....3-18
The Name Selection Menu....3-21
The Function Selection Menu....3-22
I/O Function Menus....3-26
Storing the Configuration Information....3-34
Configuration File Storage Options....3-35
General Purpose File Management....3-38
Leaving the Exit Menu....3-38
Programming with the 14752A....3-41
Statement Syntax....3-42
Fundamental Statement Descriptions....3-43
Entering 14752A Statements into the EXAMPLE Program....3-43
Storing Your Program on a Disc....3-45

Chapter 4
Program Development

Introduction....4-1
Using the Soft Front Panel....4-1
Calling the Soft Front Panel....4-2
Named Function Selection Menu....4-2
Statement Selection Menu....4-4
Front Panel Errors....4-6
Interrupts and the Soft Front Panel....4-6
Exiting the Soft Front Panel....4-7
The "DOCUMENT" Utility....4-8
Loading "DOCUMENT"....4-8
Using the Softkeys....4-9
Keyboard Control of 14752A Statements....4-10
Keyboard CALL's of 14752A Statements....4-10
Stopping Execution of a 14752A Statement....4-11
14752A System Statements....4-12
System_disable....4-12
System_enable....4-12
System_init....4-12

14752A Errors....4-13
Configuration-Time Errors....4-13
Run-Time Errors....4-13
Handling Run-Time Errors....4-22
Writing an Error Handler....4-22
Shutting Down the I/O System When an Error Occurs....4-22
Correcting Errors When They Occur....4-24
The "SET_error_print" Statement....4-24
Software Performance....4-25
I/O Throughput....4-25
Block Data Transfer Times....4-25
Interrupt Service Times....4-26
Benchmarking I/O Performance....4-27
Expanding the Number of "Names"....4-28
Disc Configurations....4-29
Disc Configuration for Program Development....4-29
Disc Configuration for Run-Only Applications....4-29
Copying Disc Files Using the "BASIC Utilities" Software....4-30
Programming Information....4-31

Chapter 5
Analog Input Functions

Introduction....5-1
69751A Analog-to Digital Converter....5-3
Menu Entries for the 69751A A/D Converter....5-3
Programming Fundamentals....5-8
Advanced Programming....5-9
69759A Analog-to-Digital Converter....5-11
Menu Entries for the 69759A A/D Converter....5-11
Programming Fundamentals....5-20
Advanced Programming....5-21
69750A Scan Controller....5-32
Menu Entries for 69750A Scan Controller....5-34
Programming Fundamentals....5-38
Advanced Programming....5-39
Scanner Function....5-43
Menu Entries for Scanner....5-44
Programming Fundamentals....5-47
Advanced Programming....5-48
3478A Digital Multimeter....5-49
Menu Entries for the 3478A Digital Multimeter....5-49
Programming Fundamentals....5-54
Advanced Programming....5-55

Chapter 6
Analog Functions

Introduction....6-1
69700A-69706A Resistance Output....6-3

Menu Entries for Resistance Output Cards....6-3
Programming Fundamentals....6-10
Advanced Programming....6-12
69709A Power Supply Control....6-13
Menu Entries for Model 69709A....6-13
Programming Fundamentals....6-22
Advanced Programming....6-24
69720A/69721A Digital-to-Analog Converter Cards....6-31
Menu Entries for the D/A Cards....6-31
Programming Fundamentals....6-36
Advanced Programming....6-37

Chapter 7
Digital Input Functions

Introduction....7-1
69770A Isolated Digital Input....7-3
Menu Entries for Model 69770A....7-3
Programming Fundamentals....7-10
Advanced Programming....7-11
69771A Digital Input/Analog Comparator....7-13
Menu Entries for Model 69771A....7-13
Programming Fundamentals....7-18
Advanced Programming....7-21

Chapter 8
Digital Output Functions

Introduction....8-1
69730A Relay Output....8-3
Menu Entries for Model 69730A....8-3
Programming Fundamentals....8-10
Advanced Programming....8-12
Using the 69730A Relay Output Card as a Matrix Switch....8-14
69731B Digital Output....8-17
Menu Entries for Model 69731B....8-17
Programming Fundamentals....8-24
Advanced Programming....8-27
Using the 69731B to Program the 14570A AC Power Controller....8-30

Chapter 9
Timebase, Interrupt, and Memory Functions

Introduction....9-1
69735A Pulse Train Output....9-3
Menu Entries for Model 69735A....9-3
Programming Fundamentals....9-10
Advanced Programming....9-13

Chapter 1
Introduction

Computer Aided Test

The primary objective of computer aided test, or CAT, is to improve overall test performance through the use of a computer. CAT applications are quite varied and often involve an evolution of test capability and program development. The Hewlett-Packard 14752A Computer Aided Test system products are the framework for complete solutions to a wide variety of CAT applications.

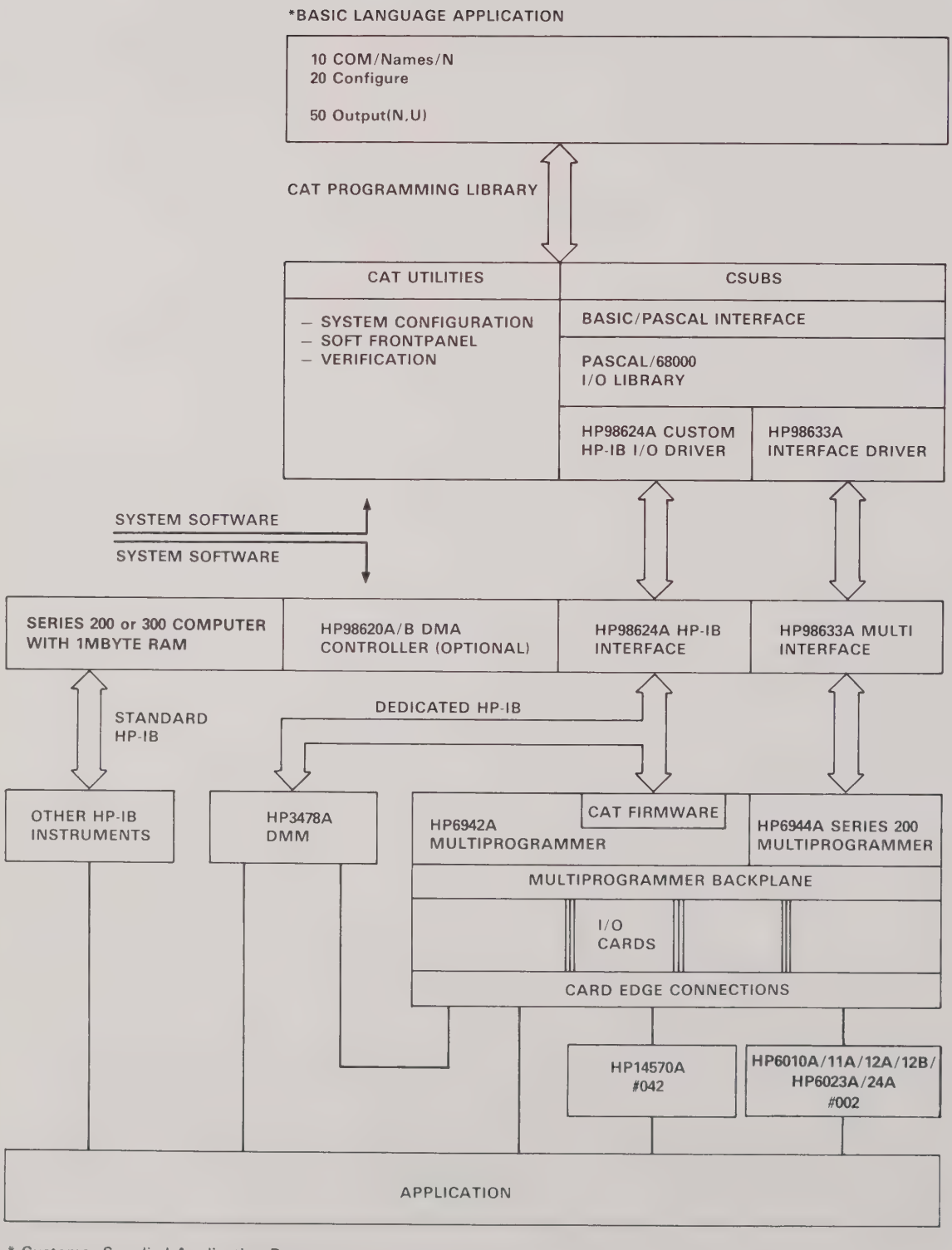
With these products you only need to concentrate on instrumenting your test and on developing your mainline applications program in the BASIC language. Program development time is significantly reduced because: I/O functions are supplied by the 14752A library, the programs you will write are self-documenting, and because hardware verification utilities are integrated into the products. Moreover, CAT performance is improved because the total solution is optimized for high-speed data transactions and for interrupt service time.

The Multiprogrammer Computer Aided Test System

The Multiprogrammer CAT solution effectively marries the flexible hardware architecture of the 6942A/6943A/6944A Multiprogrammer with the BASIC language system of Hewlett-Packard 9000 Series 200 and 300 Computers by means of the 14752A CAT Programming Package. The system is both friendly to program and fast in executing tests. The Multiprogrammer is a general-purpose device that supports a broad range of data acquisition, stimulus, and control functions by means of its plug-in I/O cards. These are easily interconnected to achieve the necessary CAT configuration. Additional Multiprogrammer functions can be incorporated into your system at any time. This modularity of test hardware is completely supported by the 14752A CAT Programming Package's software.

Figure 1-1 shows a block diagram of the 14752A CAT solution. The top half of this figure represents the system software. This is made up of the Series 200 and 300 BASIC language System, the 14752A CAT Programming Package, and your particular application program written in BASIC. The system software is discussed in greater detail in the following sections of this Chapter.

The bottom half of Figure 1-1 represents the system hardware. This consists of a Series 200 or 300 Computer connected to a Multiprogrammer mainframe. The connection to the 6944A Multiprogrammer is made with a 98633A Multiprogrammer Interface card and a 14704A, B, C Interface cable (1 meter, 2 meter or 4 meter). The connection to the 6942A Multiprogrammer is made through a dedicated 98624A HP-IB Interface. Both types of Multiprogrammers can be used at the same time. The system also supports the 3478A Digital Multimeter(s) for low level AC/DC measurements through the same dedicated HP-IB bus. A 98624A HP-IB Interface card and a HP-IB cable are needed for this purpose. Note that printers, disc drives, and other HP-IB instruments must be connected to the computer's standard HP-IB bus.



* Customer Supplied Application Program

Figure 1-1. CAT System Block Diagram

A single Multiprogrammer Mainframe will accommodate up to 16 I/O function cards; a full Multiprogrammer "link" with up to seven additional extender frames will accommodate up to 128 I/O function cards. Such a link can extend over a physical distance of up to 9 meters. The I/O function cards encompass analog inputs and outputs, digital inputs and outputs, as well as buffer memories, timing, counting, and interrupting. Consult the Multiprogrammer Series II I/O Cards Data Sheet, for a complete listing of the technical specifications and performance for each card. In addition, a Model 6010A/6011A/6012A/6012B/6023A/6024A DC Power Supply with Option 002 and the 14570A AC Power Controller with Option 042 are easily integrated into a Multiprogrammer CAT system.

The 14752A CAT Programming Package

The 14752A CAT Programming Package provides complete software support of 6942A and 6944A Multiprogrammer functions when interfaced with a Series 200 or 300 Computer. The 14752A product includes full programming documentation for use with the BASIC language system and two duplicate sets of software. This software is recorded on 3½ inch micro floppy discs for the 14752A option 044, or on 5¼ inch floppy discs for the 14752A option 042. The "master copy" of the software is meant to be stored away in a safe place as a backup for the product. The "working copy" should be used for your program development; both copies have been recorded with the identical program code (software).

The 14752A software makes extensive use of the subprogram capability of the BASIC Language System. Subprograms can be called by user supplied names of up to 15 characters. As a result, groups of short words can be strung together using the underbar character (_) to form English like, self documenting, easy-to-read programs. The 14752A software also makes extensive use of the BASIC "CSUB" capability. A CSUB is a compiled subprogram which has been written in PASCAL, but which is accessed from BASIC. Subprograms written PASCAL perform 5 to 10 times faster than their BASIC equivalents. Therefore, the I/O function calls provided in the 14752A are actually accomplished through a library of CSUBS. By using these subprograms and CSUBS, the 14752A couples the friendliness of BASIC with the performance of PASCAL.

This software product comes on four discs, three library discs and one utilities disc. One library disc contains a binary program called "INTR2_1" and a compiled Pascal subprogram Library for running in the BASIC 2.0 Language System. The second library disc contains a compiled Pascal subprogram for running in the BASIC 3.0 Language System. The third library disc contains a complied Pascal subprogram for running in the BASIC 4.0 Language System. The library discs also contain a configuration program called MENUER which is described in the following paragraphs. The utilities disc contains support utilities to the CAT Programming package. It contains a software front panel called FRONT_P, a library of hardware verification programs accessed through VERIFY called CARD_TEST, and a configuration file dump program called DOCUMENT.

The MENUER Subprogram

The MENUER subprogram makes the connection between each I/O function located at a particular Multiprogrammer slot address and the name that you have supplied for that function. You should select an I/O function name that makes sense from an applications viewpoint. The MENUER subprogram also sets the appropriate scale factor for each hardware function. Therefore, you should also choose scale factors that make sense for your application. Once the CAT hardware is configured with MENUER, it becomes a simple matter to write your application program in terms of these names and scale factors. For example, in an application where an accelerometer transducer is mounted to a structural beam, the voltage output of the transducer is fed into a 69751A A/D Converter. In the program, the user refers to the transducer-A/D converter pair as "Beam_accelerate" and takes its input directly into the program as G's of force.

Any combination of 6942A/6943A, and 6944A Multiprogrammers, 3478A Digital Multimeters, DC Power Supplies with Option 002, 14570A AC Power Controllers Option 042, and Multiprogrammer I/O Functions is easily configured using MENUER. All user-defined names are first supplied to the computer by means of a BASIC COM statement at the top of the application program. Then the MENUER subprogram is called with the 14752's Configure statement. When the program is executed, this combination of statements will lead you into the configuration menus where you can define your names and assign scale factors and other parameters for your hardware configuration.

After the configuration phase is complete, the MENUER subprogram stores your hardware configuration on the disc. It is easy to maintain several different configuration files for one CAT system. That way you can run several different tests with the same Multiprogrammer hardware.

Compiled PASCAL Subprogram I/O Library

The 14752A CAT Programming Package includes software support of single I/O card functions, subprograms for the 3478A Digital Multimeter, and multiple card menus necessary for configuring "multiple card functions". A multiple card function is a combination of several I/O cards that perform a higher level function. Note that the 14570A AC Power Controller with Option 042 is programmed by means of the 69731B Digital Output card and the DC Power Supply with Option 002 is programmed by means of a 69709A Power Supply Programming card.

Each module in the I/O library is a compiled PASCAL subprogram callable from the BASIC language through the CSUB header. The 14752A product is fully integrated with BASIC and with standard HP-IB commands to other instrumentation connected to the computer's standard HP-IB bus. Interrupts generated by Multiprogrammer I/O functions are fully compatible with the BASIC langugage ON INTR statements. The source of the Multiprogrammer interrupt is vectored directly to a handler subprogram without the user having to supply any polling routines; this saves considerable time in servicing interrupts. Errors generated within 14752A subprograms are fully compatible with the BASIC language ON ERROR RECOVER statement, thus minimizing error trapping problems.

Programming syntax for the 14752A consists of English-like subprogram calls from the BASIC application program to the compiled PASCAL subprograms. This syntax is a combination of fundamental programming statements which are generic to all I/O functions, and of advanced programming statements which relate to specific types of I/O functions. Three fundamental statements, Init(Name), Input(Name,Value), and Output(Name,Value), can be used to perform the bulk of your Multiprogrammer I/O programming.

The 14752A Programming Package supports multiple card menus that are necessary for configuring multiple card functions. The Buffer and A/D card menus are two basic multiple card menus that forward reference other card menus to form functions necessary for data acquisition. The Buffer can reference an A/D or D/A card and the A/D menu can reference a Scan Control card. A complete description of multiple card functions can be found in Chapter 10.

The complete PASCAL I/O library resides in RAM at all times. When a 14752A statement is executed, the slot address for the name is extracted from the data structure built by menuer. The proper control codes and data are transmitted to the I/O card backplane via the appropriate interface and the value is converted by the corresponding scale factor for that name and checked against any programmed limits.

The 14752A supports Direct Memory Access (DMA) data transfer for the 6944A Multiprogrammer with a 98633A Interface card. The DMA transfer is supported only for the 69759A 500 Kilohertz Analog to Digital Converter card and the 69791/69792A Memory System. For the A/D card this is done at a maximum rate of 180 kHz, and for the 69791A Memory System that is done at a minimum of 180 kHz. Note that a 98620A/B DMA Controller card must be installed in the Series 200 or 300 computer for this capability.

The 14752A CAT Programming Package includes a custom HP-IB driver program which allows the complied PASCAL subprograms to talk and listen over the dedicated HP-IB with the 6942A/6943A Multiprogrammer and the 3478A Multimeter in an efficient manner. The performance of the 6942A Multiprogrammer is 3 to 5 times faster than when using the Multiprogrammer's native instruction set with BASIC's HP-IB driver.

The Utility Programs

A software front panel for the Multiprogrammer (FRONT_P) is located on the Utility Program disc. This program can be loaded into the computer after you have loaded the 14752A CAT CSUB Library (LIBRARY); the front panel makes use of the MENUER generated data structures and the CAT CSUB Library. The soft front panel can only access Multiprogrammer hardware which has been previously configured by the MENUER subprogram.

The software front panel provides the user with quick and easy execution of any of the 14752A programming syntax for a named function; it will execute one statement at a time. First, the appropriately named function is selected from a menu of all the system's names. Second, any applicable syntax for that type of I/O function is selected from a second menu and executed. You will find the software front panel a time saving means of checking out your system's cabling connections during program development. It is also useful for learning and debugging I/O programming sequences for any user created multiple card function.

A second feature of the Utility Programs disc is the hardware verification library called VERIFY. This group of programs can be used to check the basic functionality of single I/O cards. VERIFY loads one verification program at a time from the library CARD_TEST. These verification programs, in turn, use 14752A programming syntax to exercise individual 69791A/69792A I/O functions. Menus in the verification routines lead the user through a sequence of softkeys to stimulate the I/O cards. These menus also provided the user with detailed instructions about how to sense the appropriate response on the I/O card edge through the connector assembly provided with the 14752A product.

A third feature of the Utility Programs disc is the name configuration dump program called DOCUMENT. This utility program is available so that you can easily obtain a listing of the menu configurations for each configured name in the COM statement.

14752A Specifications

Complete hardware specifications for the Multiprogrammer and supported I/O functions are found in their respective operating manuals.

I. Supported Functions

14570A AC Power Controller
Option 042: (For use with Multiprogrammer)
6010A/6011A/6012A/6012B/6023A/6024A DC Power Supply Option 002: (For use with Multiprogrammer)
69700A Resistance Output Card (unloaded)
69701A-69706A Resistance Output Cards
69709A Power Supply Control Card
69720A Voltage D/A Converter Card
69721A Current D/A Converter Card
69730A Relay Output Card
69731B Digital Output Card
69735A Pulse Train Output Card
69736A Timer/Pacer Card
69750A Scan Control/Pacer Card
69751A 33 kHz A/D Converter Card
69752A 64-Channel FET Scanner Card
69754A Relay Scanner Card
69755A 16-Channel FET Scanner Card
69759A 500 kHz A/D Converter Card
69770A Isolated Digital Input Card
69771A Digital Input/Analog Comparator Card
69775A Counter/Totalizer Card
69776A Interrupt Card
69790B 4 k Memory
69791A 64 k Word Memory
69792A Memory Expansion Card
3478A Digital Multimeter

Multiple Card Menus:

Buffer menu (references A/D or D/A card to form Buffered A/D or D/A functions)
Scanner menu (referenced by 69751A/69759A A/D card to form Scanning A/D or Buffered Scanning A/D function)
Timer (combines 69736A with 69775A and optional 69731A to pace buffered A/D or D/A functions)

II. 6942A/6943A Mulitprogrammer Mainframe Specification Summary

Number of slots: Maximum of 16 plug-in function cards per mainframe or extender. Removable rear cover provides access to card slots.

Number of Frames per I/O Subsystem: Up to seven Multiprogrammer Extenders may be placed in a chain with each 6942A mainframe.

Maximum Chain Length: A chain of Multiprogrammers and extenders can be up to 152 meters (500 feet) long.

Power Supplies: All necessary power for up to 16 plug-in functions are built into each mainframe or extender. Each frame includes three ±18 VDC supplies isolated from each other and from ground.

III. 6944A Multiprogrammer Mainframe Specifications Summary

Number of slots: Maximum of 16 plug-in function cards per 6944A mainframe. Removable rear cover provides access to card slots.

Number of Frames per I/O Subsystem: Up to eight Multiprogrammer frames may be placed in a link with each 98633A Multiprogrammer Interface card.

Maximum Link Length: A link of Multiprogrammers can be up to 9 meters long.

Power Supplies: All necessary power for up to 16 plug-in functions are built into each mainframe. Each frame includes three ±18 VDC supplies isolated from each other and from ground.

IV. Read/Write Memory Requirements

One Megabyte of RAM memory is required, as allocated below:

Memory-Resident Software	BASIC 2.0	BASIC 3.0	BASIC 4.0
RAM-based BASIC Language System	284 k bytes	266 k bytes	254 k bytes
RAM-based Language Extensions			
Extensions 2.1	175 k bytes	-	
CRTA Binary	-	4.3 k bytes	5 k bytes
CRTB Binary	-	5.4 k bytes	9.8 k bytes
DISC Binary	-	7.8 k bytes	6.8 k bytes
HPIB Binary	-	11.5 k bytes	11.5 k bytes
IO Binary	-	11.1 k bytes	10.5 k bytes
ERR Binary	-	7.3 k bytes	7.3 k bytes
KBD Binary	-	12.3 k bytes	12 k bytes
MAT Binary	-	21.2 k bytes	20.5 k bytes
CLOCK Binary	-	3.9 k bytes	3.5 k bytes
GRAPH Binary	-	41.1 k bytes	43.5 k bytes
Total language system RAM	459 k bytes	391 k bytes	385 k bytes
14752A CATPP Library (including storage for up to 128 names)	200 k bytes	210 k bytes	210 k bytes
Total resident software memory requirement	659 k bytes	601 k bytes	595 k bytes
Approximate Memory Remaining for User's application program	365 k bytes	423 k bytes	429 k bytes
Run-time Overlaid Software (added to resident software when used):			
14752A Soft Front Panel	29 k bytes	29 k bytes	29 k bytes
14752A Verification Utility	65 k bytes	65 k bytes	65 k bytes
14752A Configuration Menu	33 k bytes	33 k bytes	33 k bytes

Maximum number of Names = 512
(increase 14752A memory resident software by an additional 114 k bytes).

V. Software Performance Analysis

The time required to execute the 14752A statements comes from two factors:
- — the time required for the software to manipulate and communicate data to the hardware
- — the time required for the hardware to complete its operation

The following tables indicate just the time for the software.

Block Data Transfer Times

Multiprogrammer	6944A			6942A	
I/O Card	Inputs	DMA Inputs	Outputs	Inputs	Outputs
69751A	9 kHz	-	-	8.5 kHz	-
69759A	9 kHz	180 kHz	-	8.5 kHz	-
69770A	9 kHz	-	-	8.5 kHz	-
69771A	9 kHz	-	-	8.5 kHz	-
69720A	-	-	14 kHz	-	10.5 kHz
69721A	-	-	14 kHz	-	10.5 kHz
69731B	-	-	14 kHz	-	10.5 kHz
69790B	15 kHz	-	30 kHz	13 kHz	18 kHz
69791A	30 kHz	200 kHz*	30 kHz	13 kHz	18 kHz
69791A/69792A	30 kHz	180 kHz*	30 kHz	13 kHz	18 kHz

*See Chapter 9 for a complete list of DMA Transfer Rates.

The times listed in the following tables are based on an 8 MHz CPU as used in the Series 200 computers. Series 300 computers significantly reduce these times because the Model 310 computer has a 10 MHz CPU and the Model 320 computer has a 16.6 MHz CPU.

Block Transfer Entry Overhead

HP 6944A	:	2.5 ms
HP 6942A	:	3.0 ms

"rblock" Data Converson Time (per word)

Standard:	.45 ms per reading
User:	.45 ms per reading
None:	.20 ms per reading

V. Software Performance Analysis (continued)

The times listed in the following tables are based on an 8 MHz CPU as used in the Series 200 computers. Series 300 computers significantly reduce these times because the Model 310 computer has a 10 MHz CPU and the Model 320 computer has a 16.6 MHz CPU.

Statement Execution Time (for often used statements)

Statement	Description	Time (ms)	
		6944A	6942A
Init	Set function to a predefined state.	2.3	2.3
Input	Trigger a measurement and retrieve data.	1.4	1.9
Get	Retrieve previously triggered data.	1.3	1.4
Output	Set stimulus/control function to a value.	1.5	2.0
Preset	Preload an stimulus/control function with data.	1.3	1.4
Start	Trigger a function.	1.4	1.6
Check_done	Is the function's operation complete?	1.1	1.3
Enable_intr	Interrupt controller when done.	1.4	1.4
Disable_intr	Negate an "Enable_intr"	1.0	1.0

Corrections for Data Conversion and Limit Checking Options (applies to Init, Input, Get, Output, and Preset statements).

Condition	Effect on Execution Time
Limit Checking off	—
Limit Checking on	+.10 ms
Standard Data Conversion	—
User Data Conversion	+.35 ms
None Data Conversion	−.10 ms

Interrupt Service Response Time

Interrupt to "User_handler" entry:	13.2 ms
User_handler subprogram:	application dependent
"User_handler" SUBEXIT back to main program:	0.5 ms

14752A Documentation

Figure 1-2 shows a documentation map for the Multiprogrammer Computer Aided Test system. New users of computer aided test are encouraged to read AN 316-0 Introduction to CAT, and the BASIC Programming Techniques which is included with the Series 200 and 300 Computer BASIC Manual Kit. The Multiprogrammer Product Notes and Application Notes contain many good ideas about how to configure your particular CAT application. Full 14752A programming details are covered in the operating manuals for each I/O card.

NOTE

If it is necessary to change any factory installed jumper and switch settings for individual I/O cards, remember to load all configuration values into the appropriate configuration menus. Also, the information provided in the I/O card manuals concerning wake up code jumpers does not apply when cards are programmed with the 14752A software.

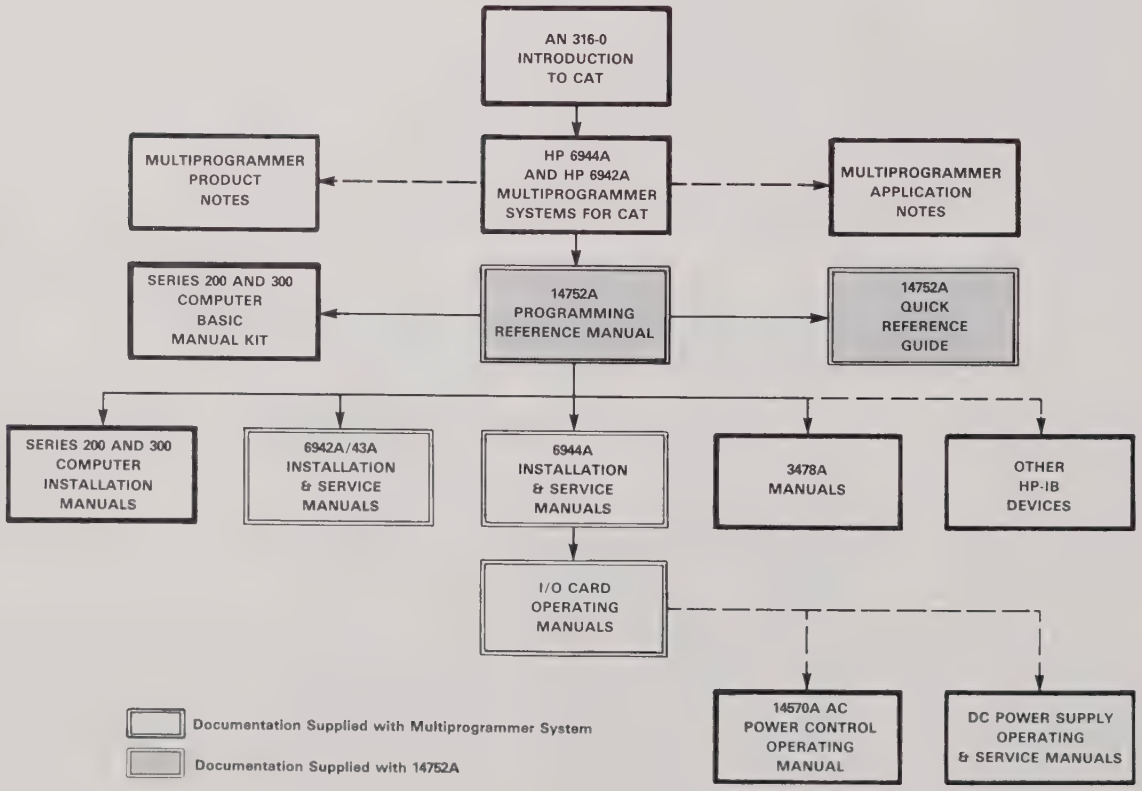

AN 316-0 INTRODUCTION TO CAT
MULTIPROGRAMMER PRODUCT NOTES
HP 6944A AND HP 6942A MULTIPROGRAMMER SYSTEMS FOR CAT
MULTIPROGRAMMER APPLICATION NOTES
SERIES 200 AND 300 COMPUTER BASIC MANUAL KIT
14752A PROGRAMMING REFERENCE MANUAL
14752A QUICK REFERENCE GUIDE
SERIES 200 AND 300 COMPUTER INSTALLATION MANUALS
6942A/43A INSTALLATION & SERVICE MANUALS
6944A INSTALLATION & SERVICE MANUALS
3478A MANUALS
OTHER HP-IB DEVICES
I/O CARD OPERATING MANUALS
14570A AC POWER CONTROL OPERATING MANUAL
DC POWER SUPPLY OPERATING & SERVICE MANUALS
Documentation Supplied with Multiprogrammer System
Documentation Supplied with 14752A

Figure 1-2. CAT System Documentation

The following list provides a brief description of the various documents as they are shown in Figure 1-2.

AN 316-0, Introduction to CAT (5952-4102)

This document introduces the terminology and fundamental concepts associated with computer aided test systems.

HP 6944A and HP 6942A Multiprogrammer Systems for Computer Aided Test (5952-4143)

This document lists and gives the specifications for the equipment you need in a computer aided test system.

Product Notes*

Some of the product notes that discuss various Multiprogrammer applications by actual users are:
- 6942A-1 Multiprogrammer Application Story in Production Line Testing (5952-4080)
- 6942A-2 Multiprogrammer Application Story in Heavy Industry (5952-4095)
- 6942A-3 Multiprogrammer Application Story in Instrument Control (5952-4096)
- 6942A-4 Multiprogrammer Application Story in Research and Development (5952-4101)
- 6942A-5 Multiprogrammer Application Story in Materials Evaluation (5952-4104)

Application Notes*

Application notes that discuss how to configure a Multiprogrammer system for specific applications are:
- AN 316-1, Buffered Analog-to-Digital Conversion (5952-4072)
- AN 316-2, Waveform Digitization (5952-4074)
- AN 316-3, High Speed FET Scanning (5952-4073)
- AN 316-4, Power Supply Programming (5952-4075)
- AN 316-5, Data Capture (5952-90001)

* The programming information provided in these Multiprogrammer documents is based on the 6942A's internal instruction set and not on the 14752A's programming statements.

14752A Programming Reference Manual (14752-90001)

This is the manual you are reading now. It provides installation instructions and describes how to program each Multiprogrammer I/O function using the menus and programming statements provided by the 14752A software. All of the hardware and software required to use the 14752A subprograms are also described. This manual does not describe how the user selects I/O functions for a specific application.

14752A Quick Reference Guide (14752-90003)

This is an alphabetical list and brief description of all 14752A programming statements. A list and description of error codes is also given.

Series 200/300 Computer BASIC Manual Kit

The following documents explain how to use the BASIC Language System.

Manual Part Numbers - BASIC 2.0

Programming Techniques with Extensions 2.1 Manual (09826-90011)
Interfacing Techniques Manual (09826-90025)
Condensed Reference (09826-90050)
Language Reference (09826-90056)

Manual Part Number	BASIC 3.0	BASIC 4.0
Programming Technique	98613-90010	98613-90011
Interfacing Technique	98613-90020	98613-90021
Graphics Technique	98613-90030	98613-90031
User's Guide	98613-90040	98613-90041
Language Reference	98613-90050	98613-90051
Utility Library	98613-90000	98613-90071

Series 200 and 300 Computer Installation Manuals

The following manuals provide detailed installation instructions for the computer.

Installation Manual for Models 226 and 236 Computers (09826-90098)
Installation Manual for the Model 216 Computer (09816-90000)
Installation Manual for the Model 220 Computer (09920-90000)
Installation Manual for the Model 217 Computer (09817-90000)
Installation Manual for the Model 237 Computer (09837-90001)
Installation Reference for Model 310/320 Computers (98561-90000)
9888A Bus Expander Installation and Service (09888-90000)
Peripheral Installation Guide for Series 200/300 Computers (97005-90000)

6942A/6943A Installation and Service Manuals*

The following two manuals provide installation, theory of operation, and assembly level service instructions for the 6942A and 6943A Multiprogrammers.

6942A Multiprogrammer Installation and Assembly Level Service Manual (06942-90006)
6943A Multiprogrammer Extender Installation and Assembly Level Service Manual (06943-90001)

6944A Installation and Service Manual

The following manuals provides installation, theory of operation, and assembly level service instructions for the 6944A Multiprogrammer and the 98633A Multiprogrammer interface.

6944A/98633A Multiprogrammer Installation and Service Manual (06944-90001)

I/O Card Operating Manuals*

One manual is shipped with each I/O card that is installed in the Multiprogrammer mainframe. Each manual contains installation, theory of operation, and configuration information for the applicable I/O card. This includes the card's block diagram, the card's edge connections, and jumper/switch setting(s).

AC Power Controller Model 14570A Operating and Service Manual (14570-90001)*

This manual gives operating and service instructions for the 14570A AC Power Controller which is supported by the 69731B Digital Output card.

DC Autoranging Power Supply Operating and Service Manuals

The following manuals contain operating and service instructions for the DC power supplies which are supported by the 69709A Power Supply Control card.

6010A Operating and Service Manual (06010-90001)
6011A Operating and Service Manual (06011-90001)
6012A Operating and Service Manual (06012-90001)
6012B Operating and Service Manual (06012-90004)
6023A Operating and Service Manual (06023-90001)
6024A Operating and Service Manual (06024-90001)

3478A Digital Multimeter Manuals

The following manuals contain operating and service instructions for the 3478A Multimeter.

3478A Multimeter Operator's Manual (03478-90003)
3478A Multimeter Service Manual (03478-90001)

Other HP-IB Devices

Documentation is provided with any other HP-IB supported device that can be incorporated into the Multiprogrammer System.

---

*The programming information provided in these Multiprogrammer documents is based on the 6942A's internal instruction set and not on the 14752A's programming statements. Consequently, all of the programming information in these documents should be disregarded by 14752A users.

Manual Organization

The information in this manual is organized as follows:

Chapter 1, Introduction: Gives product description, specifications for the 14752A, and a documentation map.

Chapter 2, Assembling the System: Lists the equipment and software required to use the 14752A Describes the system connections and BASIC Language System Software Requirements.

Chapter 3, Getting Started: Presents information to help you load the 14752A software, use the Menus to configure your hardware functions, write your own applications program, and save the configuration information along with the application program on a disc.

Chapter 4, Program Development: Explains the purpose of the various types of 14752A statements. It describes how to use the 14752A's soft front panel subprogram to execute the statements that may be used to program the functions you have selected. This chapter also provides guidance in developing and storing programs and in handling run time errors.

Chapter 5 through 9, I/O Functions: Describes each of the I/O functions (e.g. High Speed A/D Conversion) which are supported by the 14752A Programming Package. Each includes the following: functional description, hardware description, configuration menu parameters, program statement descriptions, and programming examples.

Chapter 10, Advanced Programming Concepts: Covers 14752A programming concepts that are not covered in detail in Chapters 5 through 9. Multiprogrammer interrupts and synchronization to external events are covered in detail.

Chapter 11, 14752A Statement Summary: Presents two lists of the 14752A statements. The first list presents the 14752A statements by function. The second list presents the 14752A statements in alphabetical order and also gives a brief description of them. Both of these lists show the syntax of the 14752A statements.

Appendix A: Describes how to use the Verification programs on the Utilities disc to check the operation of single I/O card functions.

Appendix B: Lists Run Time error messages and System Verification error messages.

Appendix C: Provides card interconnection diagrams for 14752A multiple card functions.

Chapter 2
Assembling the System

Introduction

This Chapter describes how to setup and interconnect your Computer, Multiprogrammer, and Digital Multimeter, for programming with the 14752A CAT Programming Package. By the end of this chapter, you will know how to assembly the system hardware and load the BASIC language system into the computer.

Initial Inspection

The hardware and software shipped with your system purchase was thoroughly tested and inspected before it left the factory. As soon as you receive it, check to make sure it has not been damaged in shipment. If it is damaged, report it to the carrier immediately and notify the nearest HP Sales and Service Office as soon as possible. Keep the original packing materials for the carrier's inspection, or in case any equipment has to be returned to Hewlett-Packard. Warranty information is printed on the inside cover of this manual. The equipment and software included with your system purchase is dependent upon the equipment and options you specified on your order. The shipping invoice lists everything that should be included. If anything is missing, report it to the nearest HP Sales Office.

Included with the 14752A package, you will find a software registration card. It should be filled out and mailed back immediately.

System Equipment

The 14752A CAT Software Package is designed to be run on the HP 9000 Series 200 and 300 Computers. There are two types of computers - integrated and modular. The integrated computer (226/236) provides all human interfaces and mass storage in one single package. The Modular computers (216, 217, 220, 237, 310, 320) are packaged so that human interfaces accessories and mass storage peripherals are separated from the computer. These accessories may include display monitor, keyboard, and mouse. The 14752A CAT Software is also supported with three versions of the BASIC Language System - BASIC 2.0, BASIC 3.0 and BASIC 4.0 to run the 14752A CAT Software with RAM based BASIC Language System, 1 megabyte of RAM is required. Table 2-1 lists a minimum set of hardware and software requirement to run the 14752A CAT Software.

Table 2-1. System Equipment Guide

HP 9000 Series 200 and 300 Computers

Computer Model	External Disc Drive	External Memory	Basic[2] Language System		
HP 216[1]	Required	768 k Byte	2.0	3.0	4.0
HP 217[1]	Required	1 M Byte		3.0	4.0
HP 220	Required	1 M Byte	2.0	3.0	4.0
HP 226	Not Required	1 M Byte	2.0	3.0	4.0
HP 236	Not Required	1 M Byte	2.0	3.0	4.0
HP 237	Required	1 M Byte		3.0	4.0
HP 310	Required	Not Required			4.0
HP 320[3]	Required	1 M Byte			4.0

1. May require the use of a HP 9888A Bus Expander.
2. BASIC extension 2.1 is required for BASIC 2.0, and some Driver and Language Extension Binaries are required for BASIC 3.0 and 4.0.
3. May require the use of a HP 98568A I/O Expander.

HP 6942A Multiprogrammer System

HP 6942A Multiprogrammer
- Option 908 Rack Mounting Kit
- Option 910 Extra Operating Manual

HP 6943A Multiprogrammer Extender
- HP 14700A Interface Kit for 1st HP 6943A
- HP 14701A Interface Kit for 2nd through 7th HP 6943A
- HP 14702A Chaining Cable
- Option 908 Rack Mounting Kit
- Option 910 Extra Operating Manual

HP 98624A Interface Card
- HP 10833A HP-IB Cable (1 m)
- HP 10833B HP-IB Cable (2 m)
- HP 10833C HP-IB Cable (4 m)
- HP 10833D HP-IB Cable (0.5 m)

HP 14752A CAT Programming Package
- Programming Reference Manual (14752-90001)
- Quick Reference Guide (14752-90003)
- Edge Connector Assembly (5060-2851)
- Option 044 Software for 3½" disc drives
- Option 042 Software for 5¼" disc drives
- Option 910 Extra Operating Manual

Table 2-1. System Equipment Guide (continued)

HP 6944A Multiprogrammer System

HP 6944A Multiprogrammer
- Option 908 Rack Mounting Kit
- Option 910 Extra Operating Manual

HP 98633A Multiprogrammer Interface Card
- HP 14704A Multiprogrammer Interface Cable (1 m)
- HP 14704B Multiprogrammer Interface Cable (2 m)
- HP 14704C Multiprogrammer Interface Cable (4 m)

HP 14752A CAT Programming Package
- Programming Reference Manual (14752-90001)
- Quick Reference Guide (14752-90003)
- Edge Connector Assembly (5060-2851)
- Option 044 Software for 3½" disc drives
- Option 042 Software for 5¼" disc drives
- Option 910 Extra Operating Manual

HP 98620 A/B DMA Controller Card*

**For rapid data transfers with the HP 69759A or HP 69791/69792A Multiprogrammer Cards.*

HP 14752A Supported Instrumentation (any or all of the following)

3478A Digital Multimeter*
6010A/6011A/6012A/6012B/6024A/6023A DC Power Supply (with Option 002)
14570A AC Power Controller (with Option 042)
69700A Resistance Output Card (unloaded)
69701A-69706A Resistance Output Card
69709A Power Supply Control Card (for Option 002 power supplies)
69720A Voltage D/A Converter Card
69721A Current D/A Converter Card
69730A Relay Output Card
69731B Digital Output Card
69735A Pulse Train Output Card
69736A Timer/Pacer Card
69750A Scan Control/Pacer Card
69751A 33 kHz A/D Converter Card
69752A 64-Channel FET Scanner Card
69754A Relay Scanner Card
69755A 16-Channel FET Scanner Card
69759A 500 kHz A/D Converter Card**
69771A Digital Input/Analog Comparator Card
69775A Counter/Totalizer Card
69776A Interrupt Card
69790B Memory Card
69791A/69792A Memory Card system**

** Requires a dedicated 98624A HP-IB Interface Card when used with a 6944A.*

*** 98620B DMA Card required for DMA transfers.*

Setting Up the System Hardware

The 14752A CAT Software can be used with the 6942A Multiprogrammer and a dedicated 98624A HP-IB Interface card, or the 6944A Multiprogrammer and its 98633A Interface Card. The CAT software requires enough read/write memory to load the BASIC Language System, the CAT software library, and your application program. If the computer does not have enough I/O slots to accomodate all of the necessary memory and the necessary interface cards; it may require the additional installation of a 9888A Bus Expander or a 98568A I/O Expander.

CAUTION

Accessories such as memory boards, interface cards, and I/O cards can be damaged by electrostatic discharge. Use the antistatic envelope when handling accessories. Don't touch the electrical traces or set the board on a charged surface (like a cloth).

CAT System Computer Installation

Your computer is shipped with an installation manual that provides complete installation procedures. Follow the installation procedures to verify that the computer is functioning properly.

On all modular computers, you should also follow the installation procedures for all non-interface type accessories such as monitors, keyboards, and disc drives.

When this is all done, you are ready to install memory and interface accessories.

CAT System Accessory Installation

Accessories are devices that insert into you computer's accessories slot. The accessories required by the CAT software are memory cards, Multiprogrammer Interface card, and the dedicated HP-IB Interface card. All accessories can be installed in any available slot, except interface accessories with attached cover plates. They must be installed in even numbered slots. The following computer installations for each model are typical slot allotment of the 14752A CAT system for the Series 200 and 300 computers.

NOTE

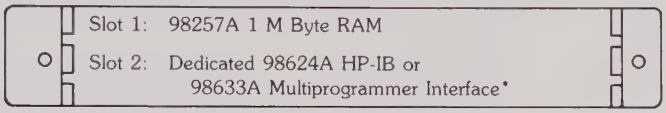
If you have more accessories than slots, you must use a 9888A Bus Expander for Series 200 computers, or a 98568A I/O Expander for Series 300 Computers which provides additional slots.

If a 9888A expander is used, the 98620A/B DMA controller ***MUST*** *be installed in the Series 200 computer's accessory slots.*

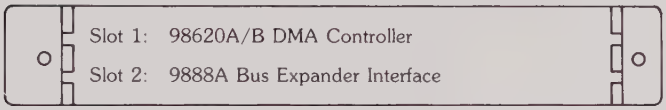
Model 216 Computer Installation

The Model 216 has two accessory slots as shown. In a minimum configuration the Model 216 can be configured as a complete CAT system. In an expanded configuration the 9888A Bus Expander must be used. In this configuration, the 9888A Bus Expander accessory slot assignment is also used.

Minimum Configuration Slot Assignment



Expanded Configuration Slot Assignment

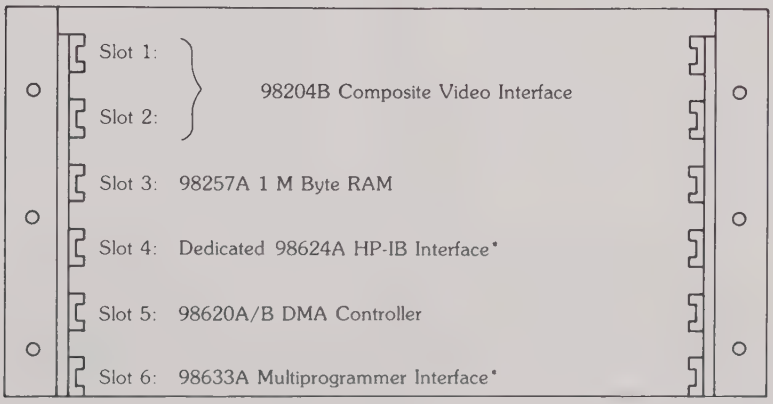


* The dedicated HP-IB interface is only required if the CAT software is used to communicate with the 6942A Multiprogrammer or a 3478A Digital Multimeter. The Multiprogrammer Interface is required when the CAT Software is used with the 6944A Multiprogrammer.

Model 217 Computer Installation

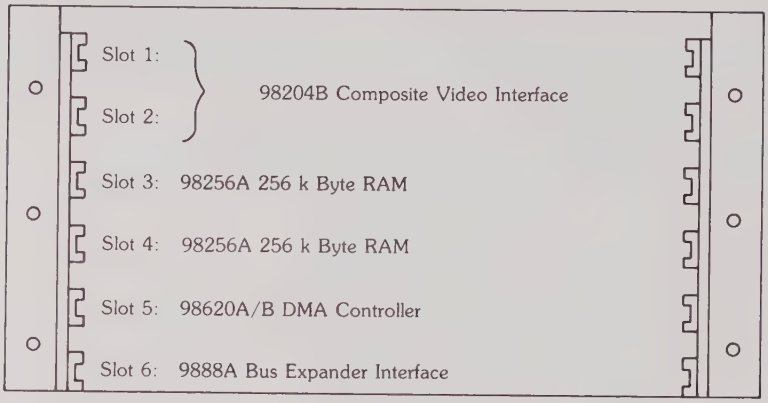
The Model 217 has six accessory slots as shown. Two of the slots are dedicated to the 98204B Composite Video Interface. The remaining slots are available for your CAT System accessories. If the 98257A 1 M Byte RAM card is used, the CAT System accessories can all be contained in the computer accessory slots. If 98256A 256 k Byte RAM cards are used, the 9888A Bus Expander must be used.

Slot Assignment Example with 98257A 1 M Byte RAM Card


Slot 1:
Slot 2:
98204B Composite Video Interface
Slot 3: 98257A 1 M Byte RAM
Slot 4: Dedicated 98624A HP-IB Interface*
Slot 5: 98620A/B DMA Controller
Slot 6: 98633A Multiprogrammer Interface*

* The dedicated HP-IB interface is only required if the CAT software is used to communicate with the 6942A Multiprogrammer or a 3478A Digital Multimeter. The Multiprogrammer Interface is required when the CAT Software is used with the 6944A Multiprogrammer.

Slot Assignment Example with 98256A 256 k Byte RAM Card

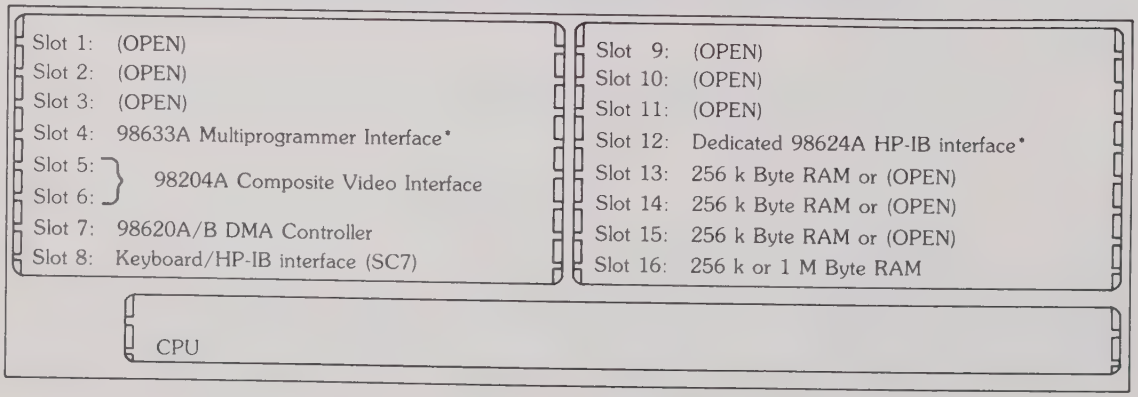

Slot 1:
Slot 2:
98204B Composite Video Interface
Slot 3: 98256A 256 k Byte RAM
Slot 4: 98256A 256 k Byte RAM
Slot 5: 98620A/B DMA Controller
Slot 6: 9888A Bus Expander Interface

* The dedicated HP-IB interface is only required if the CAT software is used to communicate with the 6942A Multiprogrammer or an 3478A Digital Multimeter. The Multiprogrammer Interface is required when the CAT software is used with the 6944A Multiprogrammer.

Model 220 Computer Installation

The Model 220 has 16 accessory slots as shown. Three of these are dedicated video display and keyboard/HP-IB (select code 7). Your CAT System accessories can be accommodated in the remaining computer accessory slots.

Slot Assignment Example for Model 220 Computer

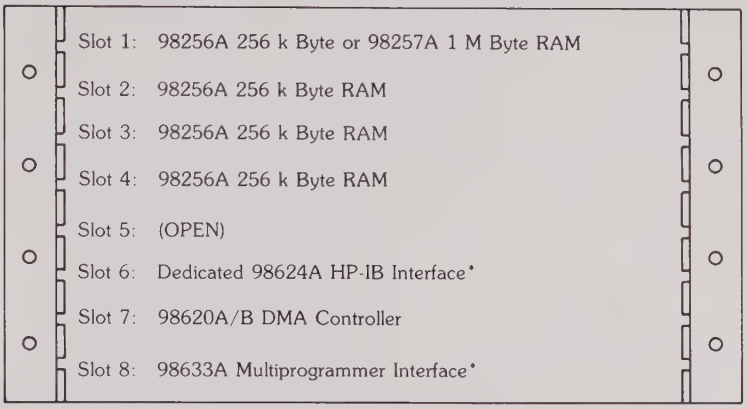

Slot 1: (OPEN)
Slot 2: (OPEN)
Slot 3: (OPEN)
Slot 4: 98633A Multiprogrammer Interface*
Slot 5:
Slot 6:
98204A Composite Video Interface
Slot 7: 98620A/B DMA Controller
Slot 8: Keyboard/HP-IB interface (SC7)
Slot 9: (OPEN)
Slot 10: (OPEN)
Slot 11: (OPEN)
Slot 12: Dedicated 98624A HP-IB interface*
Slot 13: 256 k Byte RAM or (OPEN)
Slot 14: 256 k Byte RAM or (OPEN)
Slot 15: 256 k Byte RAM or (OPEN)
Slot 16: 256 k or 1 M Byte RAM
CPU

*The dedicated HP-IB interface is only required if the CAT software is used to communicate with the 6942A Multiprogrammer or an 3478A Digital Multimeter. The Multiprogrammer Interface is required when the CAT software is used with the 6944A Multiprogrammer.

Model 226 and 236 Computer Installation

The Model 226 and 236 both have eight accessory slots as shown. The CAT System accessories can all be contained within the computer accessory slots.

Slot Assignment Example for Model 226 and 236 Computer

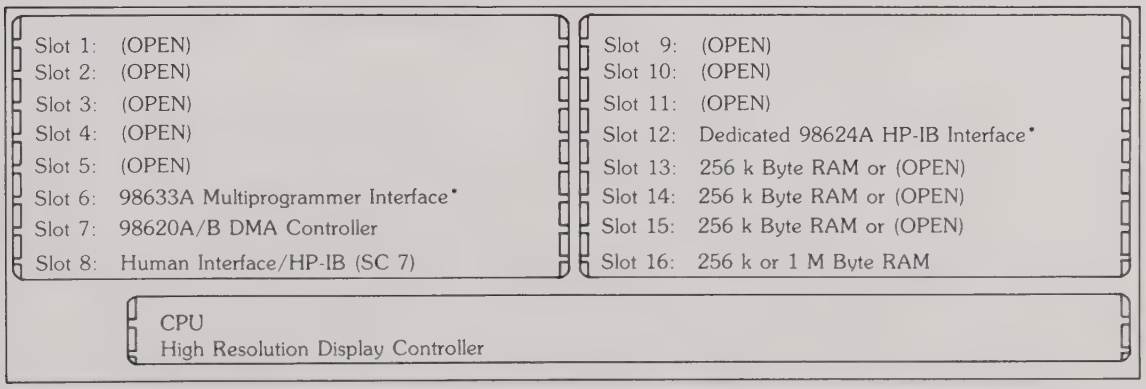

Slot 1: 98256A 256 k Byte or 98257A 1 M Byte RAM
Slot 2: 98256A 256 k Byte RAM
Slot 3: 98256A 256 k Byte RAM
Slot 4: 98256A 256 k Byte RAM
Slot 5: (OPEN)
Slot 6: Dedicated 98624A HP-IB Interface*
Slot 7: 98620A/B DMA Controller
Slot 8: 98633A Multiprogrammer Interface*

*The dedicated HP-IB interface is only required if the CAT software is used to communicate with the 6942A Multiprogrammer or an 3478A Digital Multimeter. The Multiprogrammer Interface is required when the CAT software is used with the 6944A Multiprogrammer.

Model 237 Computer Installation

The Model 237 has 16 accessory slots as shown. One of these is dedicated to the Human Interface/HP-IB interface (select code 7). Your CAT System accessories can be accommodated in the remaining computer accessory slots.

Slot Assignment Example for Model 237 Computer

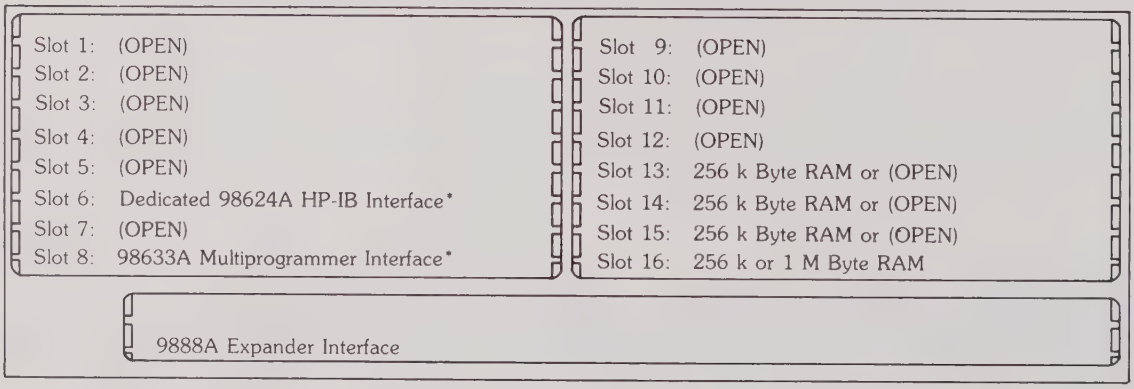

Slot 1: (OPEN)
Slot 2: (OPEN)
Slot 3: (OPEN)
Slot 4: (OPEN)
Slot 5: (OPEN)
Slot 6: 98633A Multiprogrammer Interface*
Slot 7: 98620A/B DMA Controller
Slot 8: Human Interface/HP-IB (SC 7)
Slot 9: (OPEN)
Slot 10: (OPEN)
Slot 11: (OPEN)
Slot 12: Dedicated 98624A HP-IB Interface*
Slot 13: 256 k Byte RAM or (OPEN)
Slot 14: 256 k Byte RAM or (OPEN)
Slot 15: 256 k Byte RAM or (OPEN)
Slot 16: 256 k or 1 M Byte RAM
CPU
High Resolution Display Controller

*The dedicated HP-IB interface is only required if the CAT software is used to communicate with the 6942A Multiprogrammer or an 3478A Digital Multimeter. The Multiprogrammer Interface is required when the CAT software is used with the 6944A Multiprogrammer.

9888A Bus Expander

The 9888A Expander adds an additional 16 accessory slots to your Series 200 Computer. Before installing any accessories into the expander, make sure the expander is properly installed. The 9888A Bus Expander installation manual provides complete installation procedures. Note that there are accessories which will not operate properly when installed in the expander. They include the 98625A High-speed Disc Interface, the 98620A/B DMA controller, the 98255A EPROM Interface and some ROM-based language systems such as ROM-BASIC. They must be installed in the computers accessory slots. Your CAT System accessories may reside in the expander except for the DMA Controller. Install as many accessories as possible in the computer's accessory slots.

Slot Assignment Example for the 9888A Bus Expander

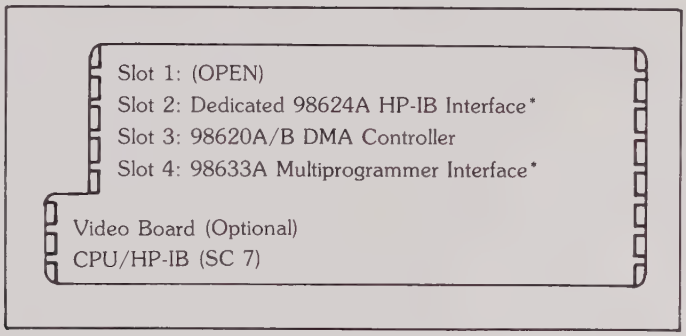

Slot 1: (OPEN)
Slot 2: (OPEN)
Slot 3: (OPEN)
Slot 4: (OPEN)
Slot 5: (OPEN)
Slot 6: Dedicated 98624A HP-IB Interface*
Slot 7: (OPEN)
Slot 8: 98633A Multiprogrammer Interface*
Slot 9: (OPEN)
Slot 10: (OPEN)
Slot 11: (OPEN)
Slot 12: (OPEN)
Slot 13: 256 k Byte RAM or (OPEN)
Slot 14: 256 k Byte RAM or (OPEN)
Slot 15: 256 k Byte RAM or (OPEN)
Slot 16: 256 k or 1 M Byte RAM
9888A Expander Interface

*The dedicated HP-IB interface is only required if the CAT software is used to communicate with the 6942A Multiprogrammer or an 3478A Digital Multimeter. The Multiprogrammer Interface is required when the CAT software is used with the 6944A Multiprogrammer.

Model 310 Computer Installation

The Model 310 has four accessory slots as shown. The CAT system accessories can all be contained within the computer accessory slots. There is sufficient internal memory in the standard Model 310 computer (1 M byte) to run the CAT software.

Slot Assignment Example for Model 310 Computer

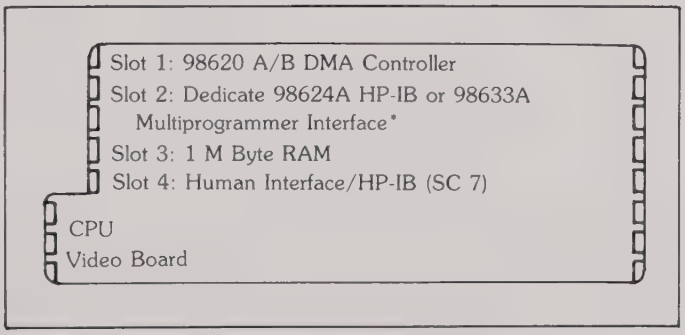

Slot 1: (OPEN)
Slot 2: Dedicated 98624A HP-IB Interface*
Slot 3: 98620A/B DMA Controller
Slot 4: 98633A Multiprogrammer Interface*
Video Board (Optional)
CPU/HP-IB (SC 7)

*The dedicated HP-IB interface is only required if the CAT software is used to communicate with the 6942A Multiprogrammer or a 3478A Digital Multimeter. The Multiprogrammer Interface is required when the CAT software is used with the 6944A Multiprogrammer.

Model 320 Computer Installation

The Model 320 has four accessory slots as shown. One of these is dedicated to the Human Interface/HP-IB Interface (select code 7). In a minimum configuration, the Model 320 can be configured as a complete CAT system. In an expanded configuration the HP 98568A /IO Expander must be used. This is the case if you have a second 1 M byte RAM card or if you are using 256 k byte RAM cards. Note that there are no restrictions on board assignments when using the HP 98568A I/O Expander.

Minimum Configuration Slot Assignment

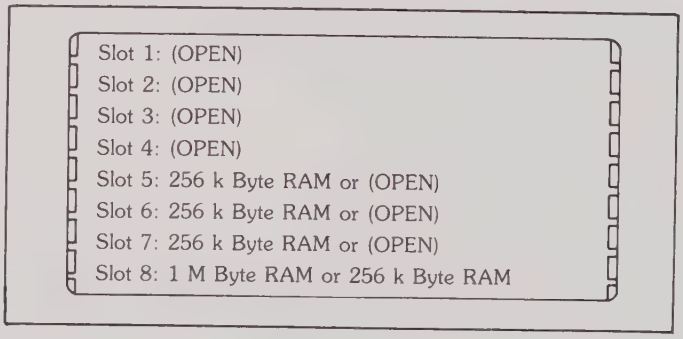

Slot 1: 98620 A/B DMA Controller
Slot 2: Dedicate 98624A HP-IB or 98633A
Multiprogrammer Interface*
Slot 3: 1 M Byte RAM
Slot 4: Human Interface/HP-IB (SC 7)
CPU
Video Board

*The dedicated HP-IB interface is only required if the CAT software is used to communicate with the 6942A Multiprogrammer or a 3478A Digital Multimeter. The Multiprogrammer Interface is required when the CAT software is used with the 6944A Multiprogrammer.

HP 98568A I/O Expander

The HP 98568A I/O Expander adds an additional eight accessory slots to your Series 300 Computer. Before installing any accessories into the expander, be sure the expander is properly installed. The Peripheral Installation Guide for Series 200/300 Computers provides installation details. Because the HP 98568A is electrically passive, all accessory boards will work properly when installed in the expander. There are no restrictions.

Slot Assignment Example for HP 98568A I/O Expander


Slot 1: (OPEN)
Slot 2: (OPEN)
Slot 3: (OPEN)
Slot 4: (OPEN)
Slot 5: 256 k Byte RAM or (OPEN)
Slot 6: 256 k Byte RAM or (OPEN)
Slot 7: 256 k Byte RAM or (OPEN)
Slot 8: 1 M Byte RAM or 256 k Byte RAM

Installing Memory Cards

To operate the 14752A CAT software, it is recommended that at least 1 M bytes of RAM be installed. This can be any combination of 98256 265 k byte cards and 98257A 1 M byte cards. Before installing additional memory cards, you may have to remove factory-installed memory card(s) and reset the address switch. Figure 2-1 shows the address switch location.

Set the address switch on the memory card as shown in Figure 2-2 and install them into the computer. Verify the amount of memory by turning on the computer and checking the Available Bytes from selftest. If it does not correspond to the amount of memory installed, switch the computer off, remove the board and check the start address switches are set correctly and the board is inserted correctly.

The 256 k bytes of internal RAM on a Model 216 computer will be counted as one 256 k bytes memory card and the address switch must be set as one of the 98256A cards.

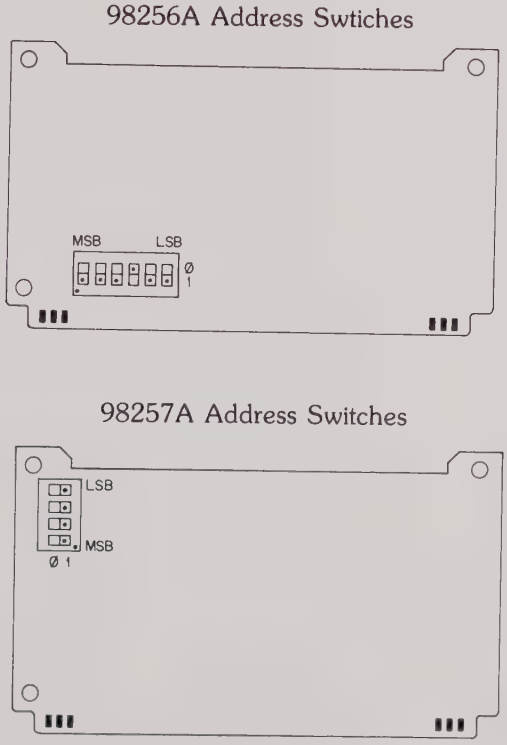


Figure 2-1. Address Switch Location

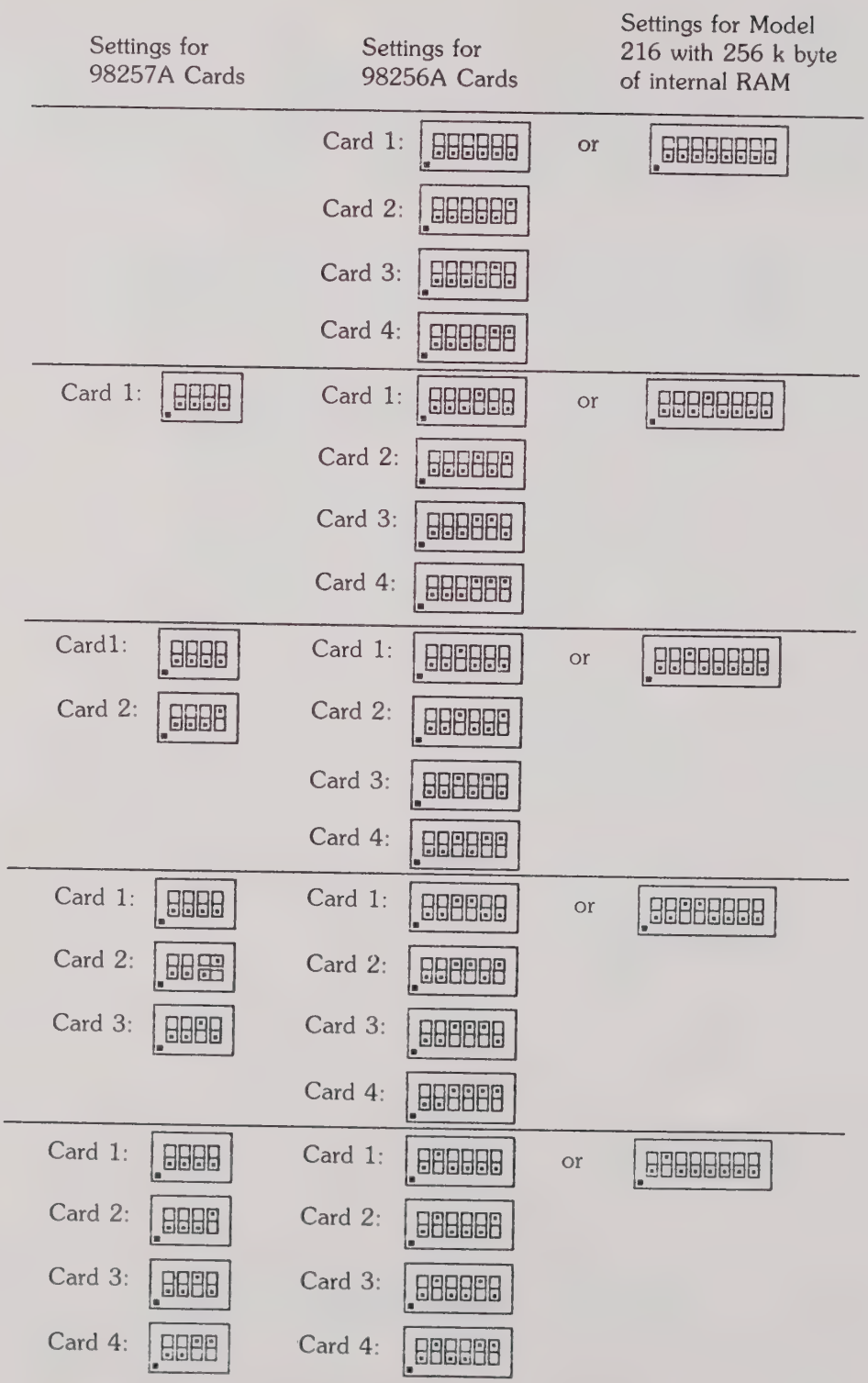
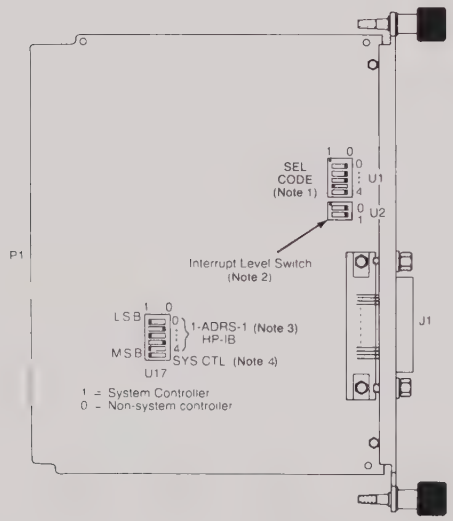


Figure 2-2. Memory Card Address Switch Settings

Installing the Dedicated 98624A HP-IB Interface Card

If you are using the 6942A Multiprogrammer or the 3478A Digital Multimeter with the CAT Software, you must also install the 98624A HP-IB Interface card in the computer. This interface will be dedicated to the CAT software and only instruments supported by the CAT software can be connected to this interface. Read the installation note that comes with the card before installing it. The 98624A is set to HP-IB select code 8, which is the default value for the CAT software. If the select code is changed to an address other than 8, the configuration menus in the CAT software must be changed accordingly as described in Chapter 3. Make sure that the interrupt level switch, address switch, and system controller switch are set as shown in Figure 2-3.



INTERRUPT LEVEL SWITCH

INTERRUPT LEVEL	SW. SETTINGS 1	0
3	0	0

NOTES
1. SELECT CODE (SEL CODE) SWITCHES SET FOR 8.
2. INTERRUPT LEVEL SWITCHES SHOULD BE SET FOR LEVEL 3.
3. ADDRESSS SWITCHES SHOULD BE SET FOR INTERNAL POWER UP PRIMARY ADDRESS OF 21.
4. THE BOARD SHOULD BE SET FOR SYSTEM CONTROLLER.

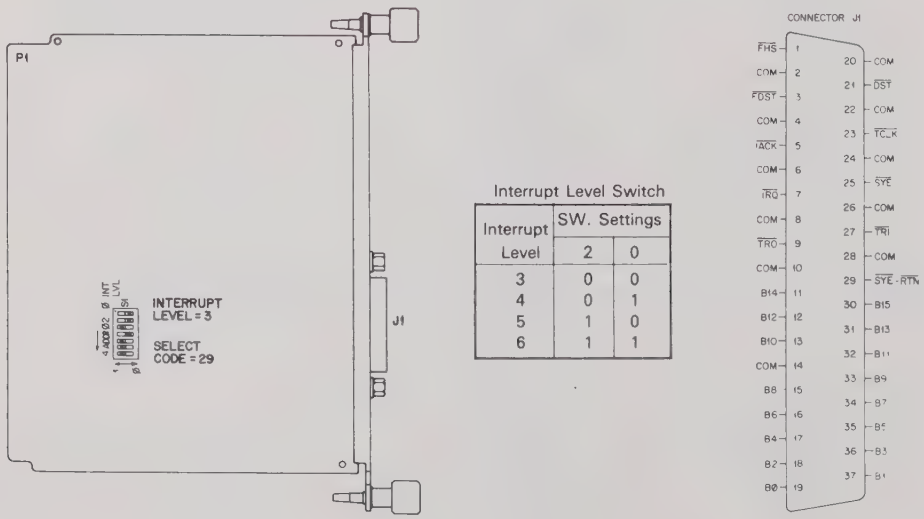
Figure 2-3. 98624A HP-IB Interface Card Switch Settings

Installing the 98633A Multiprogrammer Interface Card

The 98633A Multiprogrammer Interface card will occupy one I/O slot of the computer backplane. Read the installation note that comes with the card before installing it. Only the 6944A Multiprogrammer can be connected to the 98633A interface. Each interface card in the computer must have a unique select code. Select code 0 through 7 are reserved for the computer's internal use, leaving codes 8 through 31 for external interface cards. The 98633A is set to select code 29, which is the default value for the 14752A software. Do not set the 98633A select code to values between 0 to 7. If the select code on the Multiprogrammer Interface is changed to an address other than 29, the configuration menus in the CAT software must be changed accordingly as described in Chapter 3. The interrupt level should also be set to interrupt level three as shown in Figure 2-4.

NOTE

The 98633A Interface card occupies twice the I/O memory space that is normally allocated for each select code address in the computer. When you set the select code address on the 98633A, the next select code address above the setting is unavailable for use. For example if the 98633A is set to select code 29, select code 30 becomes unavailable for use.



Interrupt Level Switch

Interrupt Level	SW. Settings 2	0
3	0	0
4	0	1
5	1	0
6	1	1

Figure 2-4. 98633A Multiprogrammer Interface Card Switch Settings

CAT System Peripheral Installation

Peripherals are devices which connect to the interface accessories which you have just installed. They may include the 6942A and the 6944A Multiprogrammers, 3478A Digital Multimeter, disc drive, (e.g. 9121D Drive) and system printer (e.g. 2671A Printer).

System Printer Installation

If you have a system printer (e.g. 2671A), connect it to the built-in HP-IB interface (select code 7) on the rear of the computer. The printer is shipped set to HP-IB address 1, and should not be changed from this setting. The 14752A menus provide a softkey that allows you to "dump" the display to the system printer. The "dump device" HP-IB address that must be used is 701. Before turning the power on, check that the proper fuse is installed and that the line voltage switches are set for the correct line-voltage input.

Flexible Disc Drive Installation

An external flexible disc (e.g. 9121D) can be connected to the internal HP-IB port (select code 7). This is required for the Modular computers. The HP-IB address for the disc drive is factory set to zero (0). Double check the fuse and line voltage setting before turing on power to the drive.

3478A Digital Multimeter Installation

If you want to program a 3478A Digital Multimeter with the 14752A software, you must connect this instrument to a dedicated 98624A HP-IB Interface Card (select code 8). First, check your multimeter's fuse and voltage setting. Then follow the installation instructions that are given in the service manual (HP Part No. 03478-90001) supplied with it. The 3478A DMM is shipped set to HP-IB address 23. This address must be changed to HP-IB address 22 (see Figure 2-5) for programming by the 14752A. All you have to do to change the address from binary "23" to binary "22", is move switch #8 (the LSB) from logic 1 to logic 0. Do not change any other address switch settings.

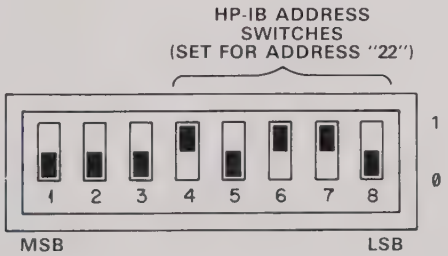


Figure 2-5. 3478A HP-IB Address Settings

HP-IB Interface Restrictions

- All the ac line switches must be turned "OFF" when connecting (and disconnecting) devices to your system.

- The total length of cable permitted on one bus system must be less than or equal to two meters times the number of devices connected together (the interface card is counted as one device).

- The total length of cable must not exceed 20 meters. For example, a system containing six devices can be connected together with cables that have a total length 12 meters (six devices x 2 m/device—12 meters). The individual lengths of cable may be distributed in any manner desired as long as the total length does exceed the allowed maximum. If more than 10 devices are to be connected together, cable shorter than two meters must be used between some of the devices to keep the total cable length less than 20 meters.

- The maximum number of devices that can be connected together in one bus system is 15. There are no restrictions to the way cables may be connected together; however, it is recommended that no more than four piggyback connectors be stacked together on one device. The resulting structure could exert enough force on the connector to damage it.

6942A/6943A Multiprogrammer Installation

Before you apply power or make your HP-IB connections to the 6942A Multiprogrammer or 6943A Extender, check the ac input power, set the HP-IB and frame address switches, and install the I/O cards as described in the following paragraphs. Rear covers are installed on the 6942A and 6943A units to hold the I/O plug-in card connector hoods in place and prevent them from loosening. These covers must be removed in order to gain access to the I/O card slots, the cable connections, and address switches. Four quarter-turn fasteners secure the cover to the rear of the unit. Figure 2-6 illustrates the rear views of the 6942A and 6943A units with the covers removed.

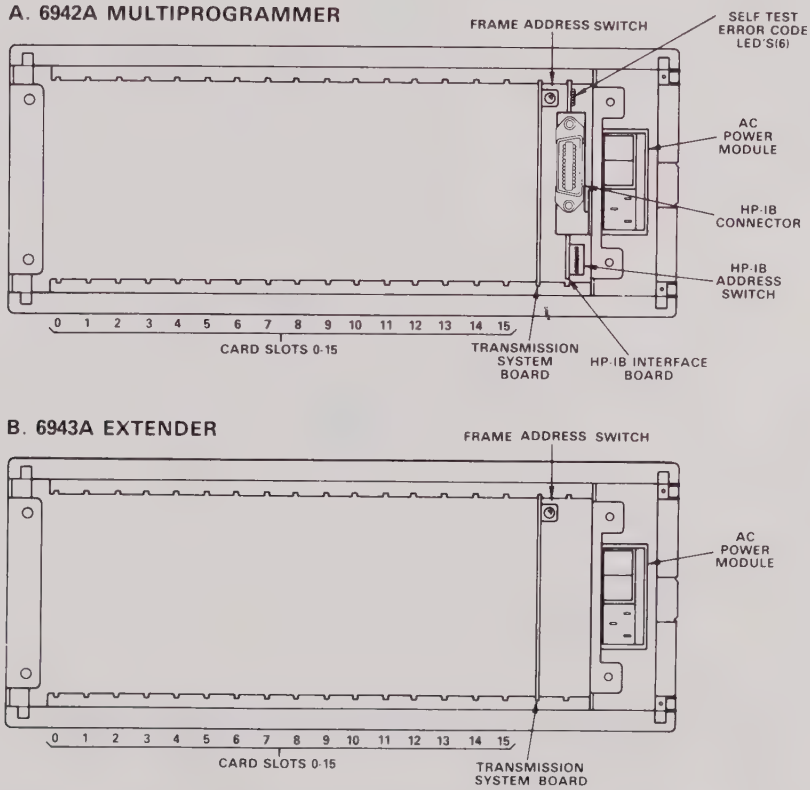

A. 6942A MULTIPROGRAMMER
FRAME ADDRESS SWITCH
SELF TEST ERROR CODE LED'S(6)
AC POWER MODULE
HP-IB CONNECTOR
HP-IB ADDRESS SWITCH
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
CARD SLOTS 0-15
TRANSMISSION SYSTEM BOARD
HP-IB INTERFACE BOARD
B. 6943A EXTENDER
FRAME ADDRESS SWITCH
AC POWER MODULE
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
CARD SLOTS 0-15
TRANSMISSION SYSTEM BOARD

Figure 2-6. 6942A and 6943A Units (Rear Views)

NOTE

The 14752A CAT Programming Package can only be used with 6942A units with serial numbers above 1950A-00320. 6942A units with serial numbers 1905A-00101 through 1950A-00320 cannot be used with the 14752A unless they are modified. If you have one of these units, contact your local HP Service Center.

1. **6942A/6943A Power Input Check.** The 6942A and 6943A unit may be operated continuously from a nominal 100 V, 120 V, or 240 V (48-63 hz) power source. A printed circuit board located with the ac power module on the rear panel (refer to Figure 2-7) selects the power source. Voltage choices are available on both sides of the PC board. Before connecting the instrument to the power source, check that the PC board selection matches the nominal line voltage of the source. The operating voltage that is selected is the one printed on the lower left side of the PC board. As shipped from the factory, the PC board in this unit is positioned for the voltage used in the area that the unit is to be delivered. To select another input voltage proceed as follows:

   a. Remove power cable from instrument.
   b. Slide plastic door on power module aside.
   c. Rotate FUSE PULL down and remove line fuse F1.
   d. Remove PC board from slot. Select operating voltage by orienting PC board to position the desired voltage on lower left side of PC board. Push board firmly into slot.
   e. Rotate FUSE PULL back up into normal position and re-insert fuse F1 in holder using caution to select the correct value for F1 (6A for 100 V or 120 V and 3 A for 220 V or 240 V).
   f. Close plastic door and connect power cable. The unit is shipped with a power cord plug that is appropriate for your location.

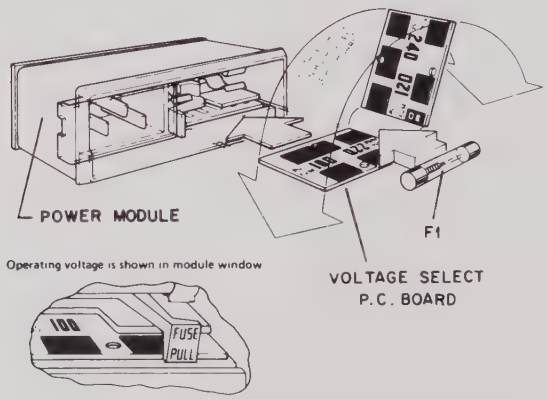


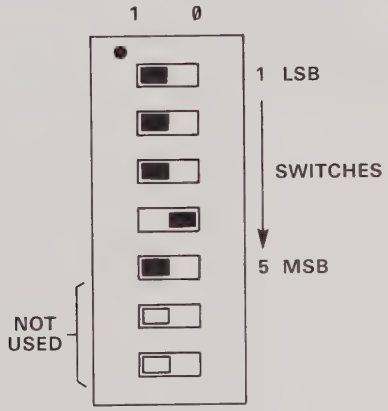
Figure 2-7. 6942A and 6943A Line Voltage Conversion

2. **Setting 6942A HP-IB Address Switches.** The HP-IB address for the Multiprogrammer system is selected by address switches on the HP-IB interface board on the back of the 6942A unit (see Figure 2-6). The switches are set for an address of 23 when the unit is shipped from the factory.

NOTE

HP-IB address 23 is the default value when programming the Multiprogrammer functions. If the address switches on the HP-IB interface board are changed to another setting (0 through 30), the function menus must be configured accordingly (refer to Chapter 3).

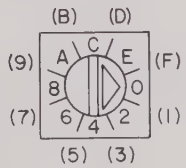
As shown in Figure 2-8 there are seven address switches on the back of the 6942A. Switches 1 through 5 are shown set to a binary representation of "23". Switches 1 through 3 and 5 are set to logic 1 while 4 is set to logic 0. The last two switches, 6 and 7, are not used. The dot on the switch assembly identifies the logic 1 side and address switch #1 (the LSB).



NOTE: SWITCH IS SHOWN SET TO ADDRESS 23

Figure 2-8. HP-IB Address Switches (Rear of 6942A)

3. **Setting Frame Address Switches on 6942A / 6943A.** A frame address switch is located on the rear of each 6942A and 6943A unit (see Figure 2-6). the frame address identifies a particular mainframe in the system. Since one 6942A Multiprogrammer and up to seven 6943A Extender units can be connected together in a Multiprogrammer system, each unit's frame address switch must be set to a different number in the range from 0 to 7. The 6942A unit is shipped from the factory with its frame address switch set to 0, while a 6943A unit is shipped with its frame address switch set to 1. You can change these settings as required (refer to Figure 2-9).



NOTE: NUMBERS/LETTERS IN PARENTHESES ARE NOT MARKED ON SWITCH. *ONLY POSITIONS 0-7 SHOULD BE USED.* SWITCH IS SHOWN SET TO FRAME ADDRESS 0

Figure 2-9. Frame Address Switch (Rear of 6942A)

NOTE

Frame address 0 is the default value for programming the Multiprogrammer functions. If the switch setting on the 6942A is changed, or if 6943A Extenders are used in the system, the function menus must be configured accordingly (refer to Chapter 3).

To change the frame address of a Multiprogrammer unit, insert a small screw driver into the switch slot and set the pointer to the desired position (0-7). In a multiple unit system (one 6942A Multiprogrammer and up to seven 6943A's Multiprogrammer extenders), the 6942A would normally be set to frame address 0 while the subsequent 6943A units would be set to frame address 1, 2, 3, etc. consecutively. Each unit in the system must have a different address.

4. **Install I/O cards in 6942A/6943A.** Install the cards and make connections from the I/O card's edge connector to your device (refer to Figure 2-10 and the applicable I/O card manual for installation details).

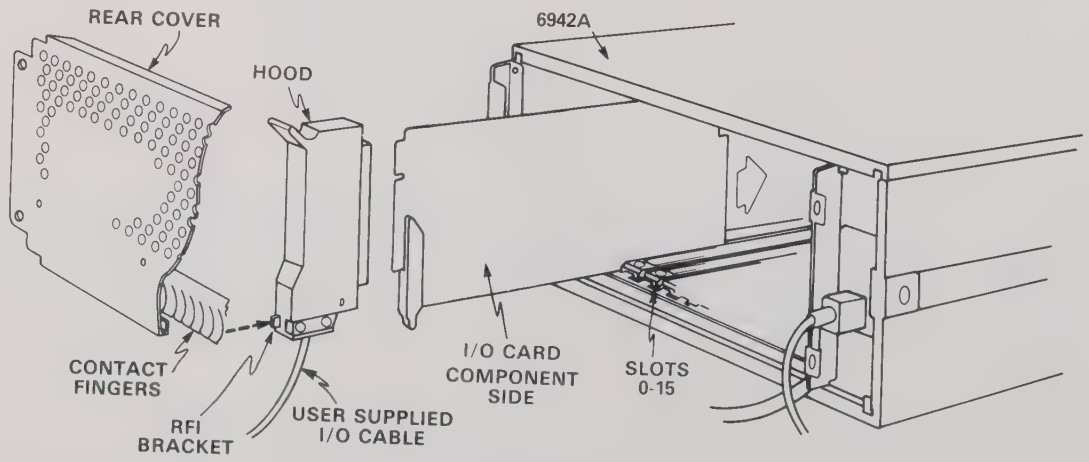
CAUTION

Always turn of power at the Multiprogrammer unit before installing or removing an I/O card. If power is not removed, it is possible to short components in the Multiprogrammer when installing or removing a card, thereby causing damage.

WARNING

Since some I/O cards can be floated up to 250 V above ground, hazardous user-applied voltage may be present on the external card edge connector. It is important to disconnect any user-applied voltages before installing or removing an I/O card.

The rear panel must be installed on the back of the 6942A after the I/O cards are installed in order to reduce EMI (electromagnetic interference) as shown in Figure 2-10. This also insures that each I/O card remains reliable seated in its backplane connector.



[1]To reduce EMI, use metal connector bracket supplied with card edge connector. (See connector assembly installation sheet, HP Part No. 14703-90001)

[2]To reduce EMI the rear cover must be securely fastened.

Figure 2-10. Installing I/O Cards in 6942A

If it is necessary to change any jumper or switch settings on these cards, remember to load all configuration values into the appropriate configuration menus. One edge connector assembly is provided with each I/O card. Assembly instructions are included with the connector. As installation and wiring are completed for each card, you should record the following information for later use during the configuration menu sequence described in Chapter 3.

- Card type and Model Number
- Frame address (0-7) in which I/O card is installed
- Slot address (0-15) of the card
- The purpose of the card in the application

6942A Multiprogrammer System Connections

The cable connections and the types of 6942A/6943A Multiprogrammer transmission system boards required depend upon the number of Multiprogrammer units used in the system. The dedicated HP-IB interface between the Multiprogrammer(s) and the computer is provided by the 98624A Interface Card. An HP-IB cable connects this interface card, which is located in the rear of the computer (or bus expander), to the HP-IB interface board in the rear of the 6942A (refer to Figure 2-11). Note that if additional devices (e.g. 3478A DVM) are to be programmed by the 14752A, they should also be connected to the dedicated HP-IB (selected code 8). These device connections can be piggy-backed onto either end of the HP-IB cable.

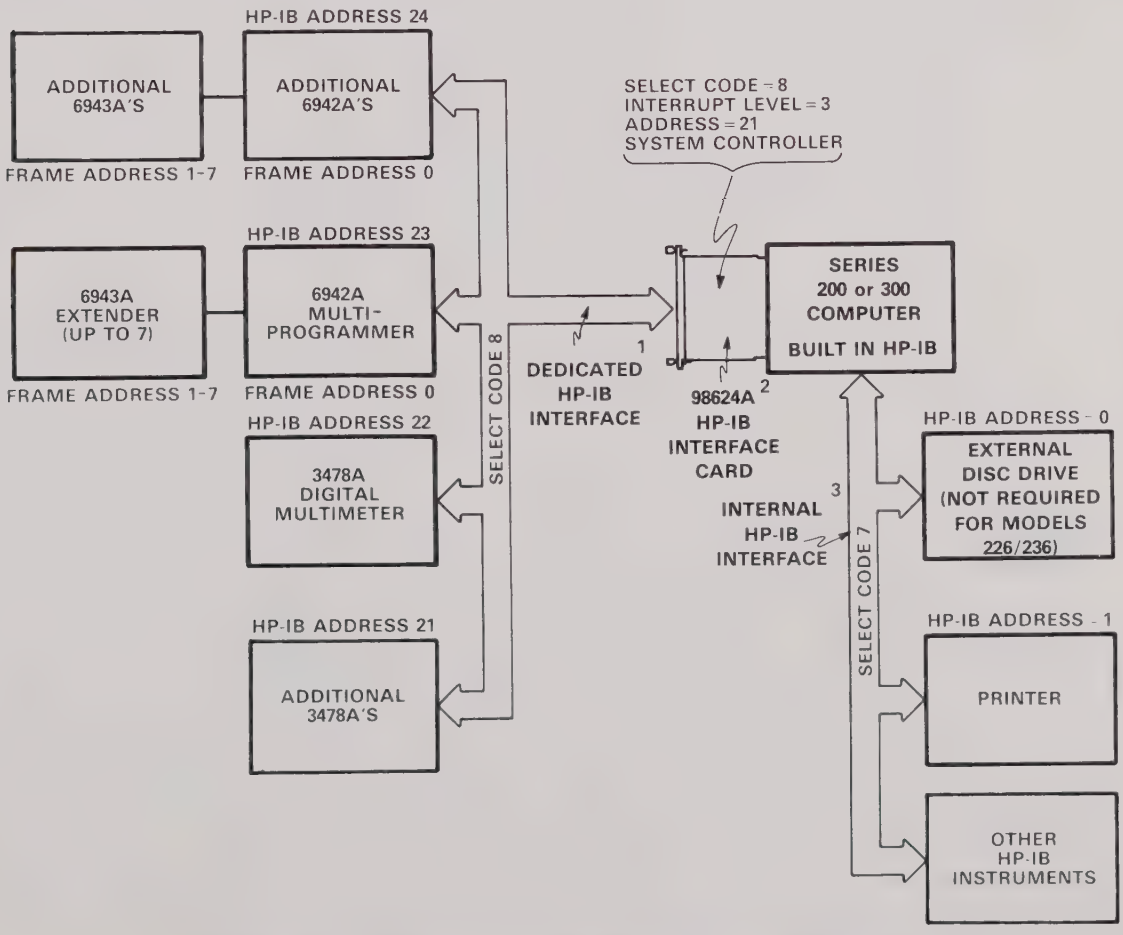


Figure 2-11. 6942A CAT System HP-IB Connections

Figure 2-12A illustrates a one unit (the 6942A Multiprogrammer) system. For this configuration, the DST Jumper Transmission System board 06942-60020 must be installed in the 6942A.

Figure 2-12B illustrates a two unit system; one 6942A unit and one 6943A unit. For this configuration, the Mainframe Transmission System board (14700-60020) must be installed in the 6942A, and the Last Extender Transmision System board (14700-60021) in the 6943A. A chaining cable (HP Part No. 14702A) interconnects the transmission boards in the two units. The maximum distance between the 6942A and 6943A is 152 meters (500 feet).

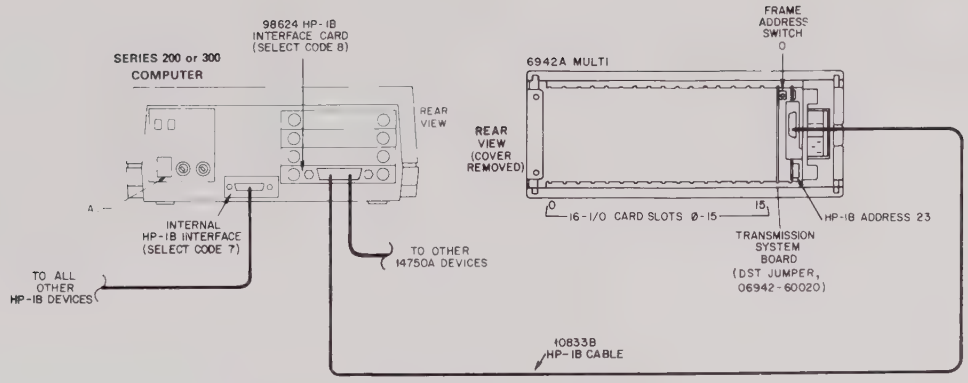
Figure 2-12C illustrates the cable connections and transmission board requirements for a system comprised of three or more (up to eight) units; one 6942A and up to seven 6943A's. The transmission board requirements for this system are as follows:

Transmission System Boards	**HP Part No.**
1 Mainframe Board	14700-60020
Up to 6 Intermediate Extender Boards	14701-60020
1 Last Extender Board	14700-60021

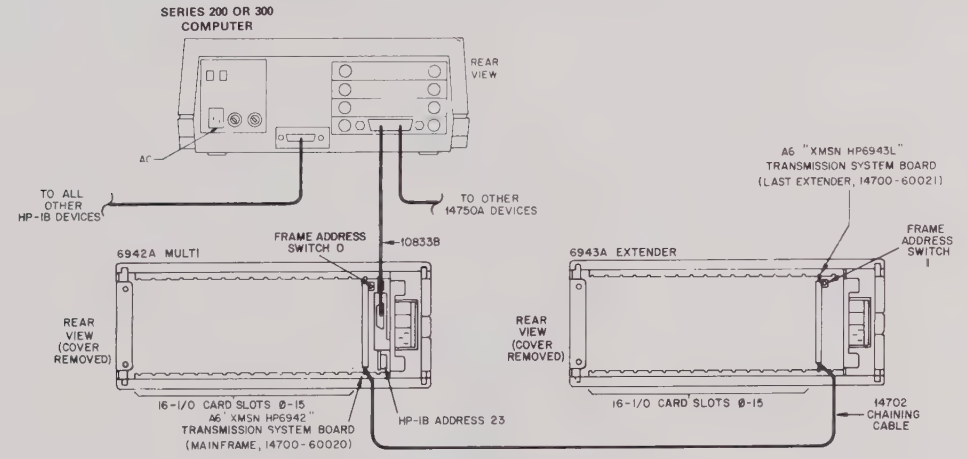
Each Intermediate Extender Transmission System board is equipped with a double connector which provides connections with the units that preceed and those that follow it. The maximum distance between the 6942A and the last 6943A is 152 meters (500 feet).

NOTE

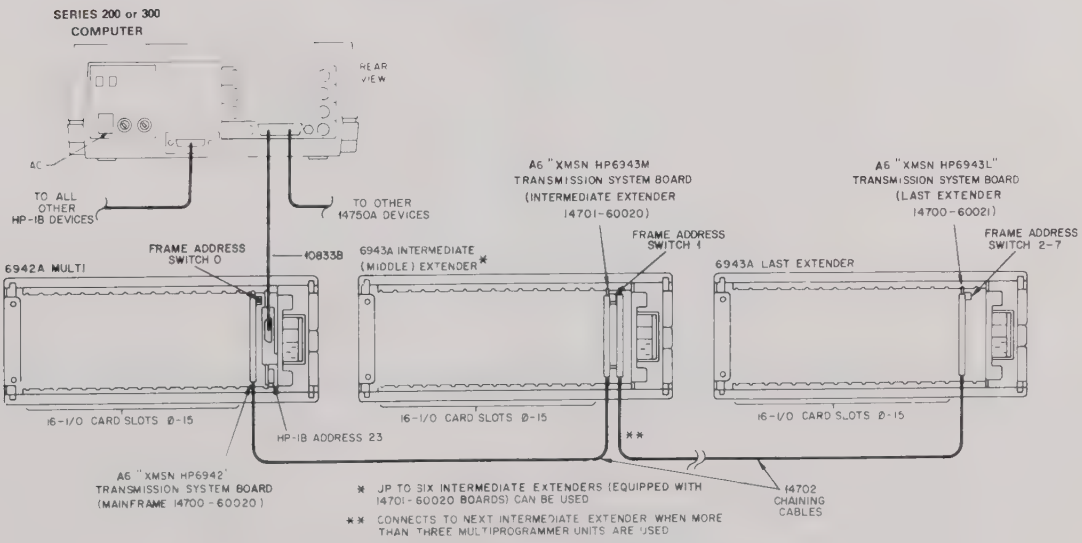
If the computer that is used in Figure 2-12 is a Model 216, the 98624A Interface Card is installed in back of the 9888A Bus Expander (see Figure 4).



A. ONE MULTIPROGRAMMER UNIT



B. TWO MULTIPROGRAMMER UNITS



C. THREE OR MORE (UP TO 8) MULTIPROGRAMMER UNITS

Figure 2-12. 6942A Multiprogrammer Cable Connections

6944A Multiprogrammer Installation

Complete installation procedures for the 6944A Multiprogrammer are provided in its installation manual (HP Part No. 06944-90001). These procedures include the following:

1. **Check the Power Input.** Disconnect the power cord and check that the proper fuses are installed (see Figure 2-13) Also check that the box checked off by the line voltage label at the ac input correspond with the line voltage at your receptable. The front panel of the unit must be opened to change the line voltage setting. Refer to the 6944A's installation manual (HP Part No. 06944-90001).

2. **Set the Frame Address Switch.** A frame address switch is located on the rear of each 6944A (see Figure 2-13) that identifies a particular unit in a link. Each link includes one 98633A Interface Card and up to eight 6944A Multiprogrammers. Therefore, each unit's frame address switch must be set to a different number in the range from 0 to 7. Each 6944A is shipped with its frame address switch set to 0. To change this setting, insert a small screwdriver into the switch slot and set the pointer to the desired position.

NOTE

If the switch setting on the 6944A is changed, the function menus must be configured accordingly (refer to Chapter 3).

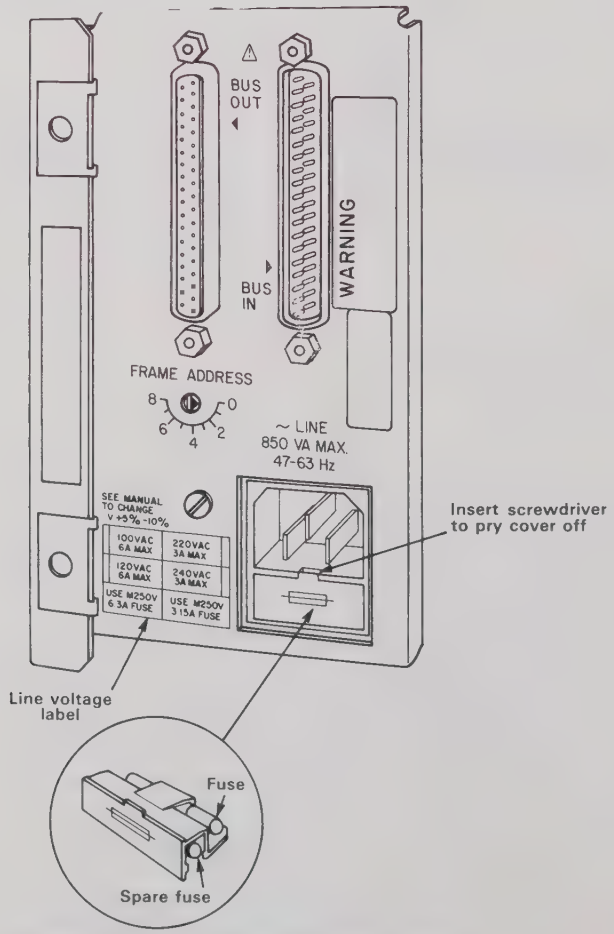


Figure 2-13. 6944A Rear Panel

3. **Check the Logic Ground Switch.** When multiple 6944A's are connected together in a link, it is recommended that a single digital ground point is used between the computer and each 98633A Multiprogrammer interface link. A logic ground switch is provided behind the front panel on each 6944A to float that unit's digital ground. Refer to the 6944A's installation manual (HP Part No. 06944-90001).

4. **Install I/O Cards in the 6944A.** Remove the rear panel from your 6944A and install the I/O cards. Make your connections from the I/O card's edge connector to your application (see Figure 2-10 and the applicable I/O card manual for installation details). One edge connector assembly with instruction is provided with each I/O card.

I/O cards can be plugged into any one of the sixteen slots without restrictions. However, there may be a power restriction regarding the total number of slots available in any particular unit when installing a 69791A/69792A Memory Card System. Refer to the 69791A/69792A's operating manual (HP Part No. 69791-90001).

The 6944A provides three +18 V power supplies for the purpose of separating analog grounds in your system. Up to 250 V can appear between separate grounds. There are restrictions as to the number of cards that can be connected to any single isolated supply. The 6944A's installation manual (HP Part No 06944-90001) contains additional information on these isolated supplies and the I/O cards that use them. Refer to the individual I/O card manuals for details about the jumper arrangements to use these isolated supplies.

CAUTION

Always turn off power at the 6944A unit before installing or removing an I/O card. If power is not removed, it is possible to damage components in the unit when installing or removing a card.

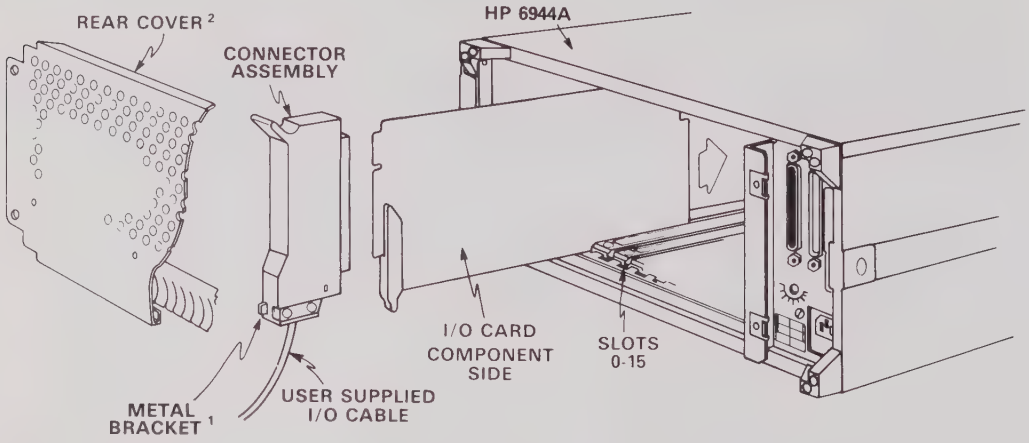
WARNING

Since isolated card input channels can be floated up to 250 V above ground, hazardous user-applied voltages may be present on the external card edge connector. It is important to disconnect any user-applied voltages before installing or removing an I/O card.

If it is necessary to change the jumper or switch settings on any card, remember to load all configuration values into the appropriate configuration menus. As installation and wiring are completed for each card, you should record the following information for later use during the configuration menu sequence described in Chapter 3.

- Card type and model number
- Frame address (0-7) in which I/O card is installed
- Slot address (0-15) of the card
- The purpose of the card in the application

The rear panel must be installed on the back of the 6944A after the I/O cards are installed in order to reduce EMI (electromagnetic interference) as shown in Figure 2-14. This also insures that each I/O card remains reliable seated in its backplane connector.

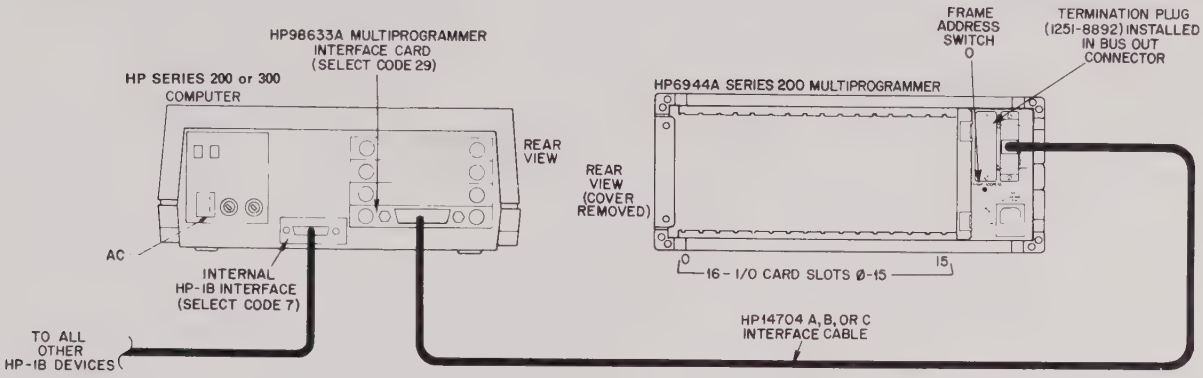


1 To reduce EMI, use metal connector bracket supplied with card edge connector. (See connector assembly installation sheet, HP Part No. 14703-90001)

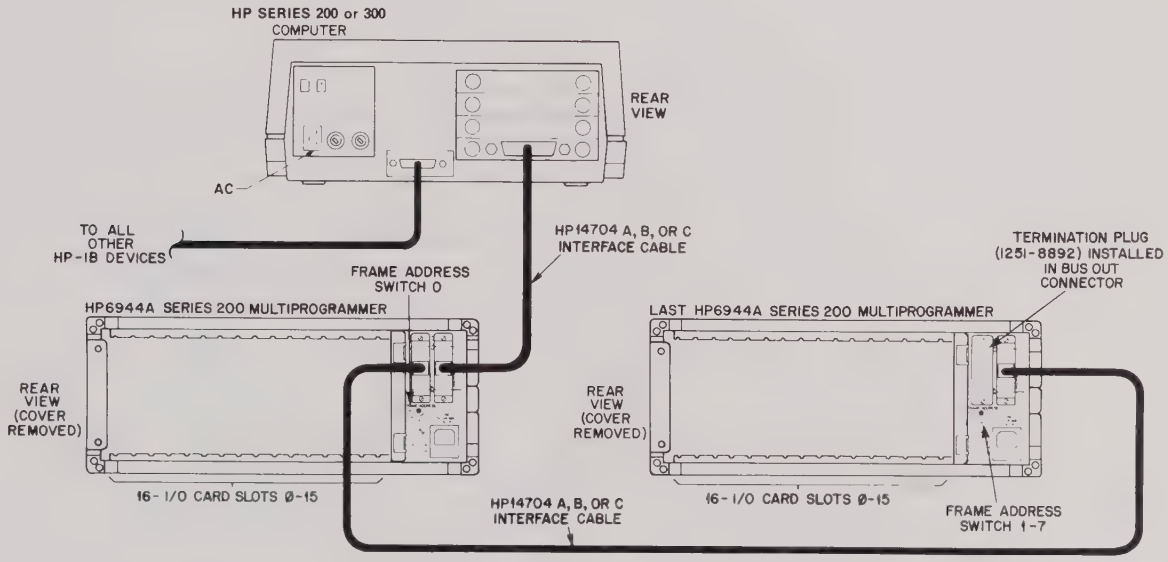
2 To reduce EMI the rear cover must be securely fastened.

Figure 2-14. Installing I/O Cards in 6944A

5. **Connect the 14704A, B, or C Multiprogrammer Interface Cable(s).** Use the 14704A, B, or C Interface Cable to connect the 98633A Interface Card in the computer to the BUS IN connector of the 6944A. Install the termination plug (HP Part No. 1251-8892) in the BUS OUT connector of the 6944A. If there is more than one 6944A in a link, install the interface cable of each additional 6944A from the BUS OUT connector on the previous unit to the BUS IN connector on the next unit in the link. The termination plug is installed in the BUS OUT connector of the last unit in the link (see Figure 2-15). Make sure that the connector plug screws are secure. The total length of all interface cabling in a 98633A Multiprogrammer interface link cannot exceed a maximum of nine meters.



A. ONE HP6944A UNIT



B. MORE THAN ONE (UP TO 8) HP6944A UNITS

Figure 2-15. 6944A Multiprogrammer Cable Connections

6944A Multiprogrammer System Connections

Figure 2-16 illustrates the system connections that can be made for a typical 6944A Multiprogrammer System. Note that all of the 6944A's in a particular link require a separate 98633A Interface Card (set to a different select code) that resides in the cardcage of the computer (or bus expander). The interface cable of the first unit in the link makes the connection to the interface card. Any standard HP-IB devices in your system such as a disc drive or a system printer, are connected to the computer's built-in HP-IB port (select code 7) with a standard HP-IB cable. Make sure that all of these devices are set to a different HP-IB address. The 3478A DMM is the only instrument that connects to the dedicated 98624A HP-IB Interface Card (select code 8) with a standard HP-IB cable. Additional 3478A DMMs can be connected to this interface provided they are set to different HP-IB addresses. HP-IB device connections can be piggybacked onto either end of an HP-IB cable.

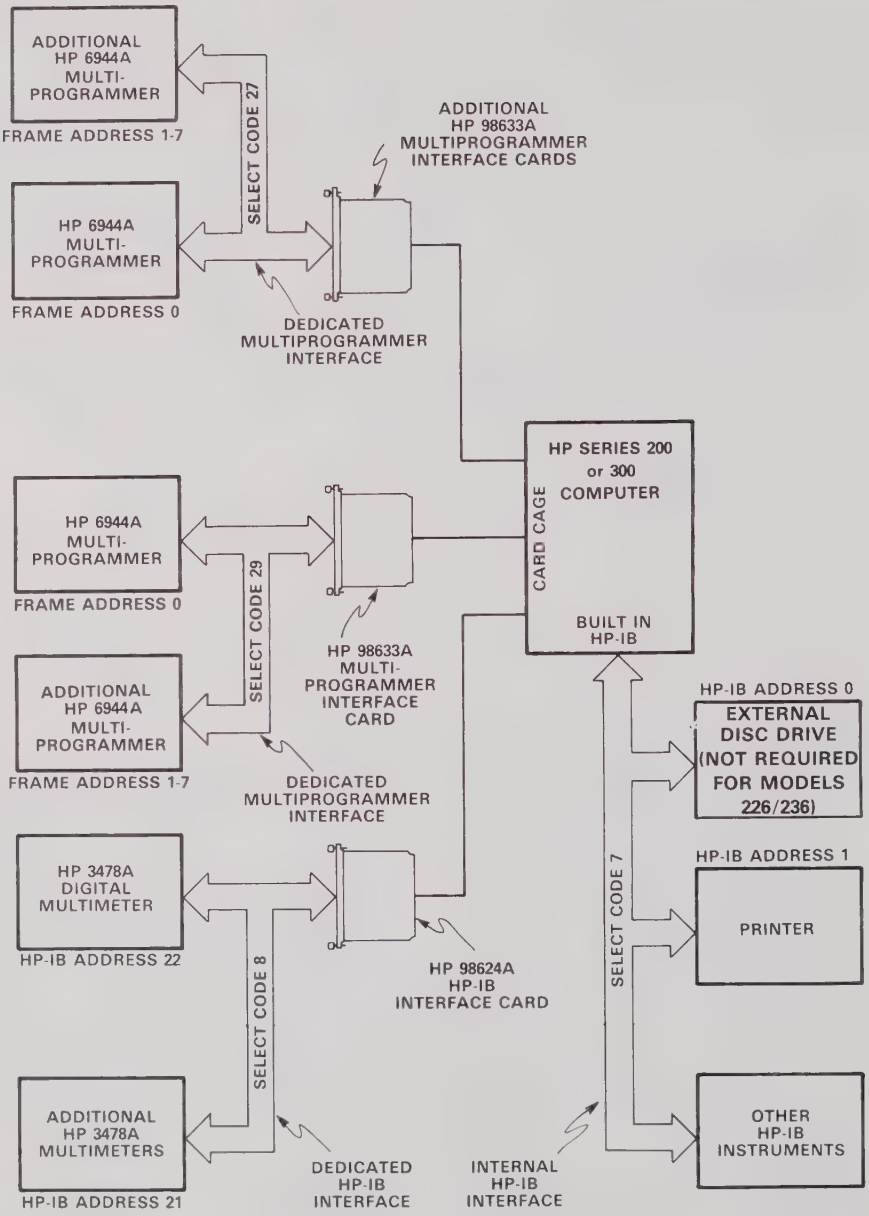


Figure 2-16. 6944A CAT System Connections

Setting Up the System Software

You are now ready to set up the BASIC language system. If you are not familiar with using or programming the BASIC language system, you may want to read the BASIC User's Guide and the BASIC Programming Techniques Manual before proceeding any further. Be sure to observe all of the disc handling precautions as follows:

- Backup discs frequently.
- Return discs to storage envelope when not in use.
- Operate your system in a clean environment.
- Maintain proper temperature and humidity.
- Avoid magnetic fields.
- Use a felt tip pen to label your disc.
- Don't bend or fold the disc.
- Don't try to clean a disc.

The following instructions tell you how to load the BASIC Language System, which is required for the 14752A CAT Programming Package. The 1472A software is supported in three versions of the BASIC Language Systems (2.0, 3.0 and 4.0). But each version of BASIC supports only a certain computer systems. The following table illustrates the compatibility of the software and hardware:

Language system	**Computer Models**
BASIC 2.0	216/226/236/220
BASIC 3.0	216/217/226/236/237/220
BASIC 4.0	216/217/226/236/237/220/310/320

Setting up System Software for Basic 2.0

1. Turn on all peripherals (e.g. Multiprogrammer, multimeter, printer, bus expander, disc drive, etc.). Don't turn on the computer yet.

2. Insert the BASIC 2.0 Language System disc in the default disc drive. Close the drive door and turn on the computer.

3. When the computer screen comes up with the message "BASIC Ready 2.0", remove the BASIC 2.0 Language System disc and insert the Extended BASIC 2.1 disc into the same drive. Close the drive door.

4. To load the Extended BASIC 2.1 routines, type:

   LOAD BIN "AP2__1"
   then press EXECUTE

NOTE

If this disc will not load, it may be because it is a version other than 2.1. If this is so, refer to the Installation Note that comes with your BASIC Extensions.

5. Extended BASIC 2.1 is loaded when the message "BASIC Ext. AP2.1" appears on the screen. Remove the Extended BASIC 2.1 disc.

NOTE

For a Model 236C, you also need to type:
LOAD BIN "GRAPH 2__1"
then press EXECUTE

6. Load the "INTR2__1" Extension

7. You are now ready to load the 14752A CAT Programming Package software as described in Chapter 3 — Getting Started.

Setting up the System Software for BASIC 3.0 or 4.0

The BASIC 3.0 or 4.0 Language System is distributed on several discs. These three BASIC discs will be used to configure a customized minimal language system for the 14752A CAT system:

BASIC System Disc
BASIC Drivers Disc
BASIC Language Extension Disc

The BASIC System Disc contains the core BASIC language system.

The BASIC Drivers Disc contains BIN files which may be required for a particular interface or product. Your hardware system may require some of these drivers.

The BASIC Language Extension Disc also contains BIN files. Each language extension BIN file provides added BASIC keywords or statements not in the core language system. The 14752A CAT Programming Package will require some of these extended language capability.

The following instruction will tell you how to set up a minimum BASIC language system for the 14752A:

1. Boot the BASIC system with the BASIC System Disc. Refer to the section on Installing or Loading BASIC in your BASIC User's Guide.

2. Replace the BASIC System Disc with the BASIC Drivers Disc in the same drive (this drive is your default disc drive.).

3. Determine the driver BIN files your system requires. Refer to the section on Configuring BASIC in your BASIC User's Guide.

NOTE

The 98633A interface does not require any driver BIN files.

4. Load the necessary driver BIN files.

5. Replace the BASIC Drivers Disc with the BASIC Language Extension Disc in the default disc drive.

6. Determine the language extension BIN files your application requires.

7. Load the language extension BIN files.

8. Load the following language extension BIN files, if not already loaded, for the 14752A software:
   - CLOCK
   - GRAPH
   - MAT
   - KBD
   - IO

   For developing software, these binary commands are recommended:
   - PDEV
   - ERR

9. You are now ready to load the 14752A CAT Programming software as described in Chapter 3 — Getting Started. You may wish to consult the Configuring BASIC section of the BASIC User's Guide for alternate methods of configuring your system.

Chapter 3
Getting Started

Introduction

Once the BASIC Language System Software is loaded into computer RAM as explained in Chapter 2, you are ready to begin using the 14752A CAT Programming Package. After reading this Chapter, you will know how to do the following:

- Load the 14752A CAT Programming Package into RAM
- Use the menus to configure a Multiprogrammer CAT System
- Save your system configuration information in a disc file
- Write a simple application program
- Save your application program in a disc file

The 14752A Software

You have been provided with two sets of operating discs. The set marked "Master Copy" should be stored in a clean, safe, dry location. This set should only be used to make a set of duplicate discs if necessary in the future. You should write protect the "Master Copy" so that you cannot inadvertently overwrite the The 14752A software. The set marked "Working Copy" can be used to store your first application program. The "Working Copy" should also be used to duplicate the discs that you program with every day. Procedures describing how to copy discs can be found in the appropriate BASIC Programming Techniques. Throughout this Chapter you will be instructed to insert various discs in the disc drive. A Model 226 computer has only a single disc drive. For other computers and disc drives, use the default disc drive.

The 14752A CAT Programming Package Disc

NOTE

There are three library discs; one for the 2.0 Language system, one for the 3.0 Language system, and one for the 4.0 Language System. Choose according to your computers language system. Part numbers are listed below.

Disc	Language System	Disc Size	HP P/N
Library	4.0	5¼"	14752-10007
Library	4.0	3½"	14752-10008
Library	3.0	5¼"	14752-10001
Library	3.0	3½"	14752-10002
Library	2.0	5¼"	14752-10005
Library	2.0	3½"	14752-10006
Utilities		5¼"	14752-10003
Utilities		3½"	14752-10004

The disc, hereafter referred to as "Library Disc" contains the programs which you will use for your CAT application. You can identify the files on a disc by inserting the disc in the disc drive, closing the door, and EXECUTing the CATalog command. This disc contains the following files:

File Name	Type	Description
MENUER	PROG	Used to create the configuration file
menu_data	BDAT	Data file for MENUER
LIBRARY	PROG	Compiled PASCAL I/O subprograms
EXAMPLE	PROG	Example application program
MY_CONFIG	BDAT	Data file for the EXAMPLE configuration
INTR2_1*	BIN	Interrupt binary for the 98633A

*2.1 Version

The EXAMPLE program is provided to help you get started quickly. You can edit this program and insert program lines for the functions that you have selected. The EXAMPLE file, which contains the "skeleton" of an application program, will be referenced in the subsequent discussions in this Chapter. The EXAMPLE file (or your own application program) must be loaded into the computer first, then all of the The 14752A subprograms in the LIBRARY file must be loaded in order to use the CAT Programming Package. The other files on the "Library Disc" are loaded automatically when needed. Throughout the discussions in this Chapter, "Library Disc" must be left in the disc drive with the door closed.

The 14752A Utility Programs Disc

This disc, hereafter referred to as "Utilities Disc", contains utility programs which will be used for verification and testing. It includes a "Soft Front Panel" subprogram which allows you to manually operate each function as if it had front panel keys. Using this feature, you can operate your function(s) without having to write a program.

The Soft Front Panel is useful in learning the The 14752A program statement associated with each function as well as in debugging your application program and checking out the cabling to the Multiprogrammer. "Utilities Disc" contains the following files:

File Name	Type	Description
VERIFY	PROG	These files contain performance verification programs which test your Multiprogrammer function(s)
CARD__TEST	PROG	
SELFT	PROG	
fp__data	BDAT	Data file for Front__p subprogram.
FRONT__P	PROG	Soft Front Panel subprogram.
DOCUMENT	PROG	Name Configuration Dump Program

The Soft Front Panel and DOCUMENT are discussed in Chapter 4; the Verification files are described in Appendix A.

NOTE

To use the BASIC ON INTR and OFF INTR statements, you ***MUST*** *load binary drivers "INTR2__1" or "IO" otherwise, a BASIC error (163) will result.*

The EXAMPLE Application Program

This program is the "skeleton" of an application program. It provides a structure which can be modified into your own application program and stored on the disc. Your program structure should follow this example. An annotated listing of the EXAMPLE is given in Figure 3-1.

As shown in Figure 3-1, a large section of the program is for interrupt and error handling. These portions are considered to be advanced programming concepts and are included for completeness; if you don't use these capabilities, the corresponding program lines can be deleted. Error handling and interrupts are discussed in Chapters 4 and 10, respectively.

The portions of a program that will be discussed in this Chapter are the basic parts that you must know in order to write a simple program using the 14752A CAT Programming Package:

1. **COM Statement**	Defines function "Names" that you wish to use in your program.
2. **Configure Statement**	A call to a BASIC subprogram which is used to configure your Multiprogrammer hardware by means of friendly menus.
3. **System_init statement**	Initializes the Multiprogrammer hardware to the configuration menus.
4. **The body of your Application Program**	Consists of a mix of BASIC statements and 14752A CAT Programming Package Statements using the Names and scale factors that you have configured to implement your test program.
5. **END Statement**	Marks the end of a BASIC main program.

Loading the EXAMPLE Program

Before loading a program from a 14752A disc, the system hardware must be set up and the BASIC Langugage System and the Extensions must be loaded into the computer RAM as described in Chapter 2. To load the EXAMPLE program, insert the 14752A CAT Programming Package disc "Library Disc" into the disc drive. Load the EXAMPLE program as follows:

Type LOAD "EXAMPLE" and then press EXECUTE

Editing the EXAMPLE Program

You can edit the EXAMPLE application program by:

Pressing EDIT and then EXECUTE

This will place the BASIC program of Figure 3-1 on the screen beginning at line number 10. At this point you can modify the program. Later in this chapter you will be instructed to insert your own program lines between lines 840 and 860. To do this you will position the cursor at line 840, press INS LN or INSERT LINE and begin to keystroke your program. At the end of each line, press the ENTER or RETURN. If you should make a typing mistake before you store (enter) the entire line, then use the cursor controls to back up to that mistake and simply overtype the correct character(s), and press ENTER or RETURN. If you should notice a mistake after storing a line, reposition the cursor to that line, make the correction and press ENTER or RETURN.

```
10    ! ========================================
20    ! =  14752A EXAMPLE APPLICATION PROGRAM   =
30    ! ========================================
40    !
50    !  All the "Names" of your I/O Functions
60    !  must be listed in the COMmon called
70    !  /Names/ here.
80    !
90    COM /Names/ Relay,Digout
100   COM /Names/ Digin,Dmm
110   !
120   !  The "Test_multi" statement below is optional.
130   !  This subprogram will self test the HP6944A at
140   !  the specified select code.  If any hardware
150   !  (mainframe or I/O card) failure is detected,
160   !  a non-zero value will be returned in the flag
      variable.                                          ─ Chapter 3
170   !
180   Test_multi(Flag,29)
190   IF Flag THEN STOP
200   !
210   !  The "Configure" subprogram interactively builds
220   !  a configuration file with information about
230   !  the I/O functions in your system.
240   !
250   Configure("Menu_on","MY_CONFIG")
260   !
270   !  All I/O functions must be initialized
280   !  before use either by "System_init"
290   !  or individual "Init(Name)" statements.
300   !
310   System_init
320   !
330   ! -------- INTERRUPT HANDLER SETUP --------
340   !
350   !  This section is used only if you will
360   !  be using the computer's interrupt
370   !  system with the 14752A software;
380   !  otherwise it should be deleted.
390   !
400   !  Set up linkage for handling interrupts
410   !  from devices on select code 29 (98633A)
420   !  and 3478's on select code 8 (98624A).
430   !  The subprogram "Intr_handler" is
440   !  part of the 14752A LIBRARY.                    ─ Chapter 10
450   !
460   ON INTR 29,3 CALL Intr_handler
470   ON INTR 8,3 CALL Intr_handler
480   !
490   !  Allow the interrupt system to
500   !  respond to interrupt requests
510   !  from the 6944A connected to
520   !  select code 29 and the 3478A
530   !  connected to select code 8.
540   !
550   Enable_sc_intr(29)
560   ENABLE INTR 8;2
```

Figure 3-1. EXAMPLE Program Listing

```
570   !
580   !  Define which I/O functions will be
590   !  allowed to interrupt, and what value
600   !  will be passed to the service routine
610   !  to identify that function.
620   !
630   Enable_intr(Digout,1)
640   Enable_intr(Digin,2)
650   Enable_intr(Dmm,3)
660   !
670   ! -------- ERROR HANDLER SETUP ------------
680   !
690   !  These next 2 statements are used only if
700   !  you want to handle run-time errors
710   !  yourself; otherwise it should be deleted.
720   !
730   !  You define what operation should take
740   !  place when an error occurs, and
750   !  disable the 14752A error messages
760   !  from being printed.
770   !
780   ON ERROR RECOVER Error_handler
790   Set_error_print(0)
800   !
810   ! ------ BODY OF APPLICATION PROGRAM ------
820   !
830   PRINT "START OF PROGRAM"
840   !
850   !  The body of your program goes here
860   !
870   PRINT "END OF PROGRAM"
880   STOP
890   !
900   ! ------- RUN-TIME ERROR HANDLER ---------
910   !
920   !  This section is used only if
930   !  you want to handle run-time errors
940   !  yourself; otherwise it should be deleted.
950   !
960 Error_handler: !
970   !
980   !  The action you wish to take when
990   !  an error occurs gets programmed here.
1000  !
1010  PRINT "ERROR OCCURRED"
1020  PRINT ERRM$
1030  STOP
1040  !
1050  ! --------- END OF MAIN PROGRAM -----------
1060  !
1070  END
1080  !
```

Chapter 10

Chapter 4

Chapter 3

Chapter 4

Chapter 3

Figure 3-1. EXAMPLE Program Listing (cont.)

```
1090  ! ========================================
1100  ! =       INTERRUPT SERVICE ROUTINE      =
1110  ! ========================================
1120  !
1130  !  This subprogram is called automatically
1140  !  by "Intr_handler" whenever an interrupt
1150  !  on select code 29 or 8 occurs.
1160  !
1170  !  It is used only if you will be using
1180  !  the computer's interrupt system
1190  !  with the 14752A software; otherwise
1200  !  it should be deleted.
1210  !
1220  SUB User_handler(Vector,Reason,Select_code)
1230  !
1240  !  This COMmon must be identical
1250  !  to that in the Main Program
1260  !
1270  COM /Names/ Relay,Digout
1280  COM /Names/ Digin,Dmm
1290  !
1300  !  Based on "Vector" (the number
1310  !  programmed in the "Enable_intr"
1320  !  statement), perform the appropriate
1330  !  interrupt service routine.
1340  !
1350  ON Vector GOTO Digout_svc,Digin_svc,Dmm_svc
1360  !
1370  ! ------ SERVICE ROUTINE FOR 'Digout' -----
1380  !
1390 Digout_svc:!
1400  !
1410  PRINT "DIGOUT INTERRUPTED"
1420  !
1430  ! 'Digout' service routine goes here
1440  !
1450  GOTO Done
1460  !
1470  ! ------ SERVICE ROUTINE FOR 'Digin' ------
1480  !
1490 Digin_svc:!
1500  !
1510  PRINT "DIGIN INTERRUPTED"
1520  !
1530  ! 'Digin' service routine goes here
1540  !
1550  GOTO Done
1560  !
1570  ! ------ SERVICE ROUTINE FOR 'Dmm' --------
1580  !
1590 Dmm_svc:!
1600  !
1610  PRINT "DMM INTERRUPTED"
1620  !
1630  ! 'Dmm' service routine goes here
1640  !
```

Chapter 10

Figure 3-1. EXAMPLE Program Listing (cont.)

```
1650  !  Further interrupts from the 3478
1660  !  "Select_code" are disabled while in
1670  !  "User_handler".  If you wish to allow
1680  !  additional interrupts to occur, they
1690  !  must be expressly re-enabled by you.
1700  !
1710  ENABLE INTR 8;2
1720  SUBEXIT
1730  !
1740  ! ----- INTERRUPT SERVICE ROUTINE EXIT ----
1750  !
1760 Done:!
1770  !
1780  !  Further interrupts from the 6944A
1790  !  "Select_code" are disabled while in
1800  !  "User_handler".  If you wish to allow
1810  !  additional interrupts to occur, they
1820  !  must be expressly re-enabled by you.
1830  !
1840  Enable_sc_intr(Select_code)
1850  SUBEND
1860  !
1870  ! ========================================
```

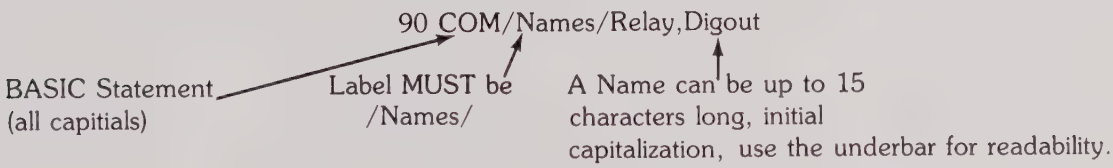
Chapter 10

Figure 3-1. EXAMPLE Program Listing (cont.)

There are two points you must watch while keystroking your program. First, BASIC keywords must be typed in all capital letters, while "Configure" and other 14752A statements **MUST** be typed in mixed mode; i.e., with the first character capital and all others lower case. This differentiates 14752A statements from BASIC keywords. For example, "Input" which is a 14752A program statement is not the same as INPUT, a keyword in the BASIC language. Second, variable names and subprogram calls are allowed to have the underbar character (_) in them. The 14752A makes liberal use of the underbar character in its statements to improve the readability of application programs.

Additional 14752A statement syntax requirements are discussed later in this Chapter and in Chapter 4. A summary of all 14752A programming statements and syntax requirements is provided in Chapter 11.

The COM Statement



The 14752A CAT Programming Package allows the user to attach one or more names to each function. An individual name, however, may only be assigned to one function. It is highly recommended that each name you choose be directly related to your application. For example, if you are using a 69730A Relay Actuator function as an activator to turn on and off a small DC motor for a pump, you might name that function "Pump_control". Or, if you have instrumented the shaft of a machine tool with a DC tachometer feeding into a 69751A A/D Converter, you could name the function "Shaft_speed". The better the choice of the name, the easier it will be to document, debug, and maintain your application program.

A labeled COM statement(s) with all of your function names must be located at the beginning of your program. The label for this COM statement **MUST** be /Names/. The order in which your names appear in the COM statement will be the same order in which these names are listed on the Name Selection Menu discussed later in the Configuration Phase discussion. Each name can be up to 15 characters long. A name must be typed in mixed mode, the first character being alphabetical and capitalized. The underbar (_) and numeric characters are allowed to be embedded within the name. Names are separated by commas.

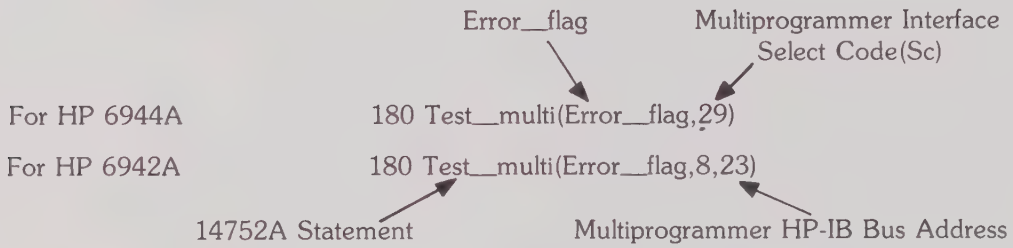
COM statements are shown in the EXAMPLE program in lines 90 and 100. Multiple program lines may be used to accommodate all of the names. Each numbered continuation line for the COM statements must also include the label /Names/. Up to 512 names can be assigned in one application program. Note, however, that if you use more than 128 names, you must perform a modification to the 14752A software as described in the discussion entitled "Expansion" in Chapter 4. Any subprograms which use some or all of these names must have an exact duplication of the COM statement(s) within the subprogram body as shown in the EXAMPLE program in lines 1270 and 1280.

NOTE

The I/O function names listed in the COM statement must be used only as the "Name" parameter in 14752A programming statements. Improper operation will result if you attempt to use these names in any other manner in your program.

You can edit the EXAMPLE program later to delete the sample names given in the COM statements and enter the names that you have selected for your particular application. Each name that you enter will be defined later during the Configuration Phase.

Test_multi Statement



The Test_multi statement is an optional statement used to test the Multiprogrammer System and its I/O cards. This statement should be used before the configure statement. Three parameters are allowed in the Test_multi syntax. To specify the 6942A selftest all three parameters are required- the Error_flag, the select code, and the bus address. However, the syntax specifying the 6944A selftest includes only two parameters- the Error_flag, and the select code. Each parameter is described in the following paragraphs.

The Error_flag is a return variable indicating the number of errors detected during the test. The Error_flag must be a REAL variable. Results are sent to the display.

The select code parameter can be a value parameter specifying the interface select code of the Multiprogrammer System. The default select code of a 98633A interface is 29 and 98624A/B interface is 8.

The bus address represents the HP-IB Interface address. If the bus address parameter is specified, then the select code is assumed to be the select code of a HP-IB Interface and not that of a 98633A. If not specified then it is assumed to be the select code of the 98633A.

The selftest tests the mainframe and I/O cards. During the mainframe portions of the test, the display for test pass or test fail is:

```
HP6944A Selftest at Select Code nn
Passed test.

HP6942A Selftest at HP-IB nnn
Passed test.
```

```
HP6944A Selftest at Select Code nn
Failed test: error-nn.

HP6942A Selftest at HP-IB nnn
Failed test: error-nn.
```

After displaying results of selftest the status of the I/O cards is displayed in the following format:

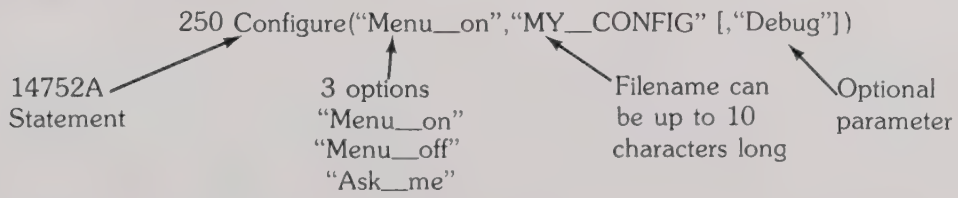
```
Frame Slot Description                    Status
```

The I/O card of each frame and slot will appear under each heading. If there is no card in the slot or the card is a 69752A, 69754A, or 69755A there will be no entry on the display for that specific slot address. If the card fails the test, the number (−52 to −61) of the error code will appear under the status column.

NOTE (for the 6942A)

If the 6942A passes the mainframe portion of the selftest, the CAT firmware program will be down loaded to the 6942A and activated at the end of the Test_multi statement.

The Configure Statement


250 Configure("Menu_on","MY_CONFIG" [,"Debug"])
14752A Statement
3 options "Menu_on" "Menu_off" "Ask_me"
Filename can be up to 10 characters long
Optional parameter

The Configure statement is a 14752A subprogram call. When it is executed when the Menu_on option, it presents you with a series of friendly menus which will be described later in the Configuration Phase section of this Chapter. The menus are used to configure the Multiprogrammer and the 3478A Digital Multimeter hardware. "Configure" means that you tell the computer exactly what I/O hardware is in your application, where it is located, and how you want to use it in

your application. The Configure statement must occur in your program **after** the COM/Names/ statement but **before** the first line of your application program. You can use Configure with different filenames to configure the same hardware functions for several different applications. However, only one file name is allowed in a Configuration statement and only one Configure statement is normally used per application program. The Configure statement in the EXAMPLE program uses the file name "MY_CONFIG". This file name will be used later as an example in the Configuration Phase discussion. The optional parameter, Debug, is used for hardware verification purposes. This allows programming of a system which failed hardware checks performed by configure (refer to Appendix A).

1. Configure("Menu_on","MY_CONFIG") → The Menu_on option must be executed the first time a program is developed and can be used to modify existing programs. Executing the Configuration statement with the Menu_on option presents you with the menus so you can define your hardware (see Configuration Phase discussion).

2. Configure("Menu_off","MY_CONFIG") → After you have filled in the menus correctly for your hardware configuration, you should change the Configure statement to Menu_off. This statement loads the configuration information into computer memory, but bypasses the menus. Menu_off should be used after the program is totally debugged and you are ready run the program.

3. Configure("Ask_me","MY_CONFIG") → During program development you may have occasion to go back and forth between modifying menus and not wanting to see menus. The Ask_me option can be used with a simple "Yes" or "No" answer to recall the menus without having to edit your Configure statement each time to type in "Menu_on" or Menu_off".

Executing the configuration statement with the "Ask_me" option causes the following prompt to appear on the computer screen with the associated softkeys.

Go through the Configuration Menus?

Pressing "Yes", presents you with the menus so you can define or re-define names associated with your hardware.

Pressing "No", causes the program to continue without interacting with the menus.

The System_init Statement

```
310 System_init
```

This statement disables I/O function interrupts and initializes all output functions in the system at the states specified in the menus during the Configuration Phase. All of the names in the COM statement are sequentially initialized by the System_init statement; or each function can be initialized individually using the Init_(Name) statement. For example, to initialize the Relay function only, change the program line to read:

```
310 Init(Relay)
```

It is good practice to initialize every I/O function in your system to the beginning of your application program. You cannot program an uninitialized function, if you attempt to do so, an error message will appear on the screen. The following is a sample error message.

```
14752A: Use of unINITialized name
        detected by Output
        for Model 69730A Relay Actuator
```

The User's Application Program

Lines 840 through 860 of the EXAMPLE program show where your application program should go. Additional lines are easily added as required depending upon the size of your program. If you write a large program, you will want to store it on a separate disc as described in Chapter 4. Your first short programs are easily stored on the same disc as the 14752A CAT Programming Package ("Library Disc").

Programs are written using the 14752A programming statements with the Names and scale factors that you have defined during the Configuration Phase. The fundamental programming statements that can be used such as Output(Name,Value) and Input(Name,Value) are described in general later in this Chapter. Example programs are provided for each supported I/O function in Chapters 5 through 9.

The END Statement

Line 1070 is the END statement. All BASIC programs must have an END statement.

Error and Interrupt Handlers

These program sections (lines 330 to 800, 900 to 1040, and 1080 to 1870) can be deleted if you do not wish to use interrupts in your system and/or do not wish to handle run-time errors in the program. Errors and interrupt handling are described in detail in Chapters 4 and 10, respectively. Both Error and Interrupt handling are provided in the EXAMPLE application program to show you the recommended program structure. Note that the Error_handler in this EXAMPLE is a subroutine which comes before the END statement and is activated by the "ON ERROR RECOVER" BASIC statement. The Interrupt Handler is a subprogram which **must** always be named "User_handler",and must come after the END statement. It is activated by the "ON INTR" BASIC statement.

Storing the EXAMPLE Application Program

At this point, if you have made changes to the EXAMPLE application program, you could store it on the disc under a different name. If you want to do this at this time, make sure that the disc on which you wish to store the program is inserted into the disc drive; then store the new program as follows:

Type STORE "FILENAME" then press EXECUTE

Refer to the section entitled "Storing Your Program On a Disc" at the end of this Chapter for additional instructions on Storing and Re-storing programs.

Loading the 14752A Library and Running EXAMPLE

The file "LIBRARY" the "Library Disc" contains the memory resident portion of the 14752A CAT Programming Package. Before you can run the EXAMPLE or any other CAT application program, the LIBRARY must be loaded into computer memory as follows:

Type LOADSUB ALL FROM "LIBRARY" then press EXECUTE or RETURN.

Note that if you have any trouble, a BASIC RESET (Shifted PAUSE key, or shifted BREAK) will always stop the program and allow you to enter another keyboard command. Since the EXAMPLE program includes the Menu_on option in the Configure statement, you will enter the Configuration Phase soon after you press the RUN key and pass Test_multi. The Configuration is described in the next section of this Chapter. Before getting the first menu in the Configuration Phase, you will notice different messages appearing on the top line of the computer display: e.g., "Loading Configuration File". Several messages will appear on the display while the system is searching for and loading various disc files into memory. If you have edited the File Name in your Configure statement and if that File Name does not exist on the particular disc which is inserted into the drive, then the following display (see Figure 3-2) will appear on the computer screen. If this message does appear, read the following, otherwise please go on ahead to the Configuration Phase discussion.

NOTE

There are two possible sets of softkeys. These are differentiated by the shaded and unshaded areas in the menu. Choose the representation applicable to your computer.

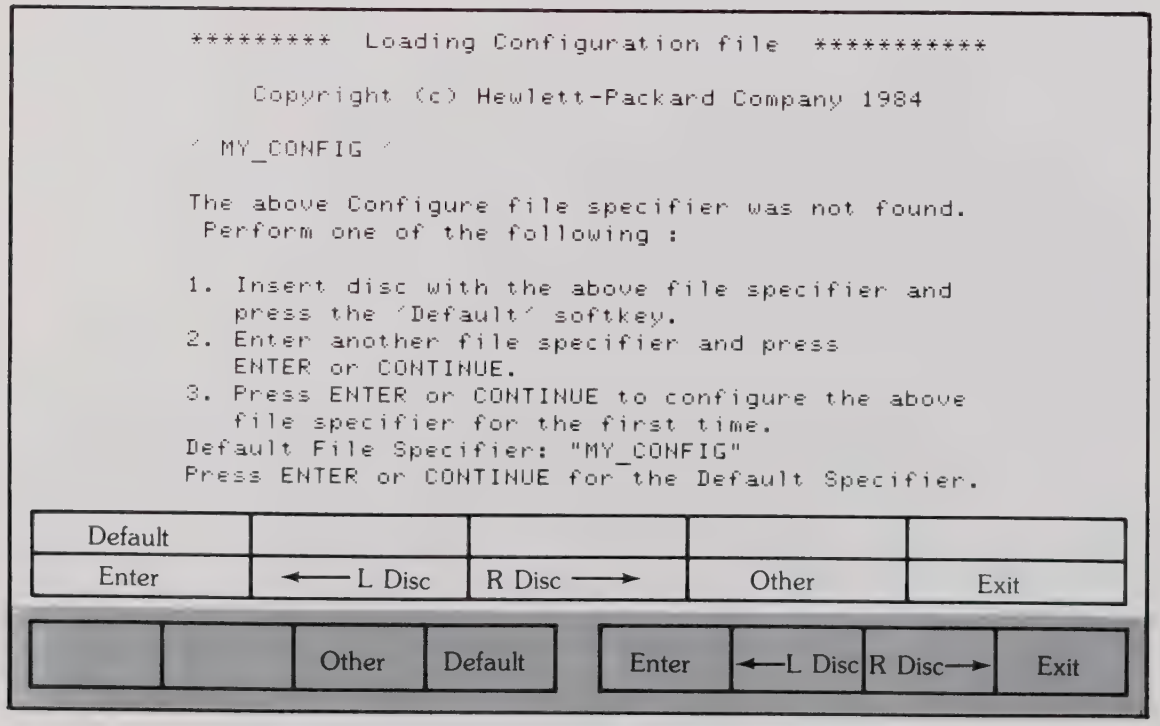

********** Loading Configuration file ***********
Copyright (c) Hewlett-Packard Company 1984
' MY_CONFIG '
The above Configure file specifier was not found.
Perform one of the following :
1. Insert disc with the above file specifier and press the 'Default' softkey.
2. Enter another file specifier and press ENTER or CONTINUE.
3. Press ENTER or CONTINUE to configure the above file specifier for the first time.
Default File Specifier: "MY_CONFIG"
Press ENTER or CONTINUE for the Default Specifier.
Default
Enter
L Disc
R Disc
Other
Exit
Other
Default
Enter
L Disc
R Disc
Exit

Figure 3-2. Configuration File Not Found Screen

Case No. 1 (Figure 3-2) The Named configuration file is on another disc: Insert the proper disc and press Default .

NOTE

In addition to file name, the literal form of the file specifier is a string expression containing an optional protect code, device type, device selector, and unit number. See the BASIC Language Reference Manual.

Case No. 2 (Figure 3-2) You want to specify another file name and/or mass storage device: Use one of the softkeys described below. Press one of these softkeys and enter the information in the area specified on the screen; then press ENTER or CONTINUE to complete the operation and access the configuration menus.

- L Disc: Use this softkey to specify another file name on the left disc drive .
- Internal: Use this softkey to specify another file name on the internal disc drive (226 computer).
- R Disc: Use this softkey to specify another file name on the right disc drive.
- Other: Use this softkey to specify another file name on a mass storage device other than the standard disc drives.

Case No.3 (Figure 3-2) You are configuring this file name for the first time: Press ENTER or CONTINUE to access the menus.

The Configuration Phase

The 14752A CAT Programming Package supports a very flexible arrangement of Multiprogrammer I/O functions and 3478A Digital Multimeter functions. One of the first tasks is communicating the hardware configuration to the computer. This is accomplished by filling in the blanks on a series of friendly menus. As long as you do not change the configuration, this step need only be performed once.

As described previously, the Configuration Phase is entered by executing the Configure statement with the "Menu_on" option specified or by replying "Yes" to the Configure "Ask_me" option.

Menu Sequence

Throughout the Configuration Phase, you will be presented with a series of menu screens. The "Welcome Menu" is the first menu that you will see. The menus that follow are the "Name Selection Menu", the "Function Selection Menu", and the menus for the particular functions that you select.

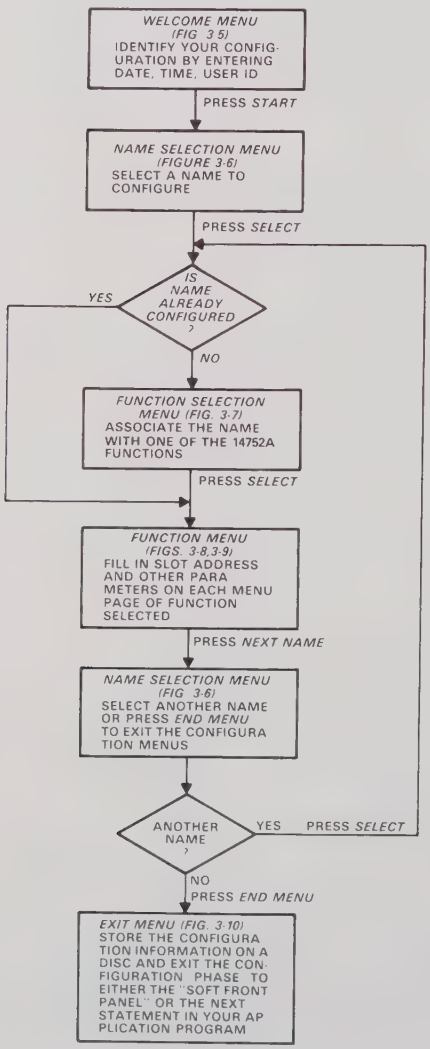

WELCOME MENU (FIG. 3-5) IDENTIFY YOUR CONFIGURATION BY ENTERING DATE, TIME, USER ID
PRESS START
NAME SELECTION MENU (FIGURE 3-6) SELECT A NAME TO CONFIGURE
PRESS SELECT
IS NAME ALREADY CONFIGURED ?
YES
NO
FUNCTION SELECTION MENU (FIG. 3-7) ASSOCIATE THE NAME WITH ONE OF THE 14752A FUNCTIONS
PRESS SELECT
FUNCTION MENU (FIGS. 3-8,3-9) FILL IN SLOT ADDRESS AND OTHER PARAMETERS ON EACH MENU PAGE OF FUNCTION SELECTED
PRESS NEXT NAME
NAME SELECTION MENU (FIG. 3-6) SELECT ANOTHER NAME OR PRESS END MENU TO EXIT THE CONFIGURATION MENUS
ANOTHER NAME ?
YES
PRESS SELECT
NO
PRESS END MENU
EXIT MENU (FIG. 3-10) STORE THE CONFIGURATION INFORMATION ON A DISC AND EXIT THE CONFIGURATION PHASE TO EITHER THE "SOFT FRONT PANEL" OR THE NEXT STATEMENT IN YOUR APPLICATION PROGRAM

Figure 3-3. Configuration Sequence

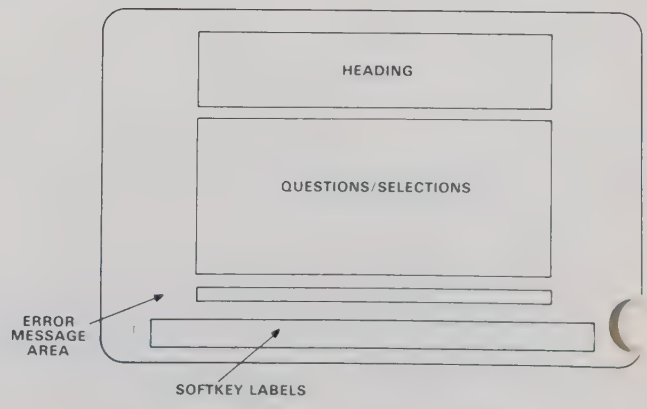


Figure 3-4. Menu Screen Areas

Figure 3-3 shows the sequence in which you will see the menus. It also summarizes the entries you must make and the menu softkeys you must press to go from menu to menu. As shown in Figure 3-3, after you have completed filling out the menus for each of your named functions, you must return to the Name Selection Menu and press the "End Menu" softkey. Pressing "End Menu" will cause the Exit Menu softkeys (Figure 3-11) to appear on the screen. These softkeys are used to store your configuration information on a disc. This procedure will be described after the various menu pages are described.

The Menu Screens

The menu screens are divided into areas as shown in Figure 3-4.

Heading: On the Welcome menu, the heading identifies the product. On the menu pages that follow the Welcome Menu, the heading contains user identification and date/time information.

Middle Area: This section of the menu contains questions to answer or selections to make. Most of the questions (or selections) have "default" answers which will be used if you do not supply an answer.

Error Message Area: If you mistakenly answer with values that are outside the allowable range, the computer will beep, ignore your input, and an error message will appear in this area on the screen. Some examples of error messages are:

"Numeric entry only"
"Legal length of entry is X"
"Field may not be passed until an entry is made"
"Illegal limit value"
"Illegal slot specifier"
"Illegal frame specifier"

When defining a multiple card function, if the front-end card name; i.e., the name of the card being referenced by the Buffer menu or A/D card menu,has not yet been configured or has not been configured as the front-end card type, an error message will appear on the screen. This message will serve as a warning. It will allow you to continue and either configure the name or supply the correct name. Some error messages of this type are:

"A name can not reference itself"
"*Name* has not yet been configured"
"A name must be supplied"
"*Name* is not *front-end type* model"

where *Name* is the function name and *front-end type* is the model number of the front-end card (i.e., 69720A, 69721A, 69751A, 69759A, or 69750A).

Softkeys. Softkey labels appear across the bottom of each menu screen. They are used in selecting and answering questions on the menu and in moving from screen to screen.

The Welcome Menu

Following the preliminary messages that appear on the screen for loading files and the "dots" that appear when the variables in the "COM/Names/" statement are initialized, the Welcome menu will appear on the screen. This menu allows you to identify each configuration by filling in the date and time as well as some identifier of your choice. This information will be included in a heading at the top of all menu pages that become part of this particular configuration file. Thus, the menu pages associated with each configuration file will be identified and time stamped.

Figure 3-5 shows the layout of this menu. At the top of the screen is a heading which identifies the product, its revision number, and its copyright dates. The middle of the screen is a presentation of several questions to be filled in by the user. At the very bottom of the screen, the softkey labels are shown. If you are unfamiliar with the use of softkeys, you might want to review this operation in the "Getting Started" Chapter of the BASIC Programming Techniques Manual.

Softkeys and Keyboard Controls. The softkeys shown on the menu screen, the knob (cursor or mouse wheel), up/down arrows, and ENTER key on the computer keyboard will assist you in selecting questions to answer, in editing your answers, in moving about from screen to screen, and in exiting from the configuration menus. On the 216 computer and the 220 computer with the 98203A keyboard, softkeys k0 through k4 are shifted softkeys. Every set of softkeys includes a Help softkey. This softkey may be selected at any time should you forget the meaning of the other keystrokes.

Help — Toggles a Help screen on and off. The Help screen provides a brief description of the other softkeys and screen controls. To return to your place in the menus press Help again.

Several keys as well as the Knob (cursor wheel) will assist you in positioning the cursor to select the question which you are about to answer.

Now try each of the following Cursor control keys. Remember, you cannot damage the equipment in any way while you are learning its operation.

Knob/Mouse (cursor wheel) — Rotating the knob advances the cursor through the list of questions in either direction. A question to be answered is selected when the cursor is stopped at that question.

Up/Down (arrows) — These cursor control keys advance the cursor down or up one question at a time in much the same way as the knob or mouse.

The funtion's entire configuration can be "dumped" to your printer at HP-IB Address 701 (see Chapter 2) by pressing the Dump softkey.

Dump — Pressing this softkey will copy the entire configuration for the current function to your line printer. If a line printer is not present in your system, the computer will beep and display an error message. This error will not halt the Configuration phase.

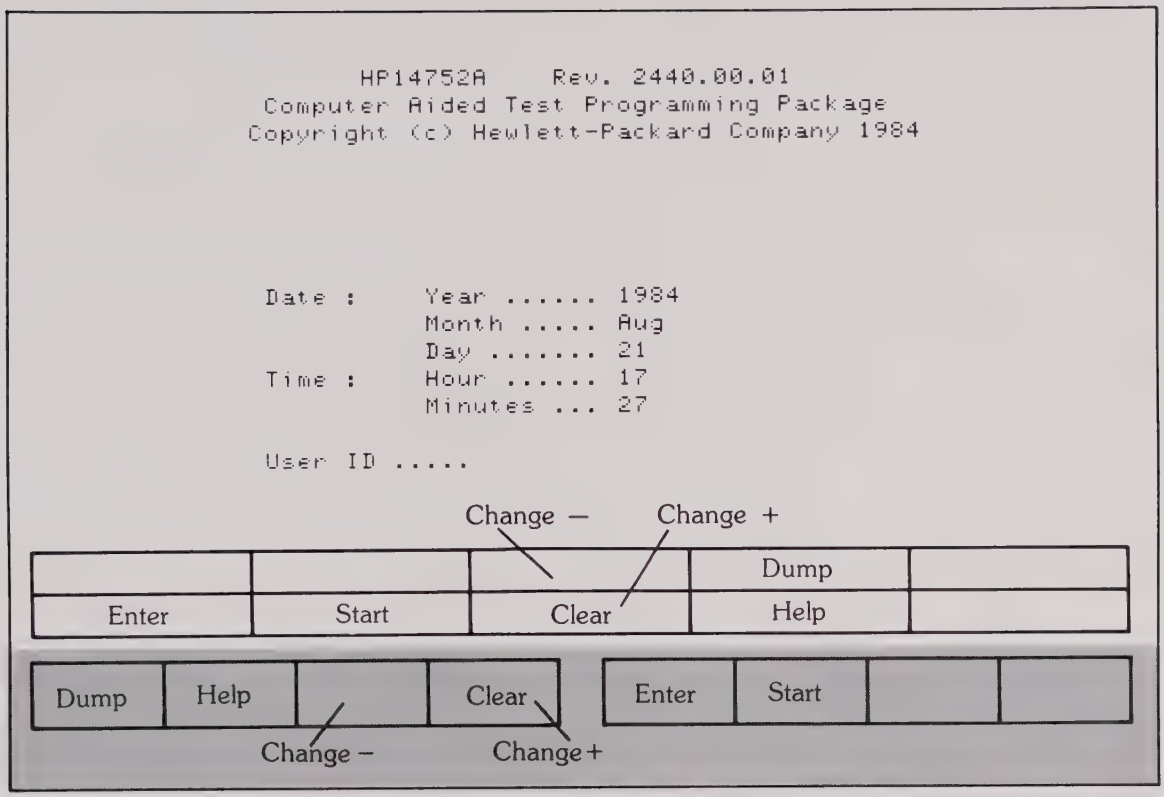


Figure 3-5. The Welcome Menu

The following keys are used when answering the questions:

ENTER or RETURN on keyboard — Pressing this key will load your response into the computer. The cursor will automatically move down one question.

Enter — Same as the keyboard ENTER or RETURN.

Clear — Used to clear any keyboard entry. For example, if you make a mistake typing in the Year or User ID entry, you can clear it by pressing the Clear softkey and typing in the entry again.

Change – and **Change +** — Used to select the alphanumeric month entry on the Welcome Menu. These keys appear in the menu only when the cursor is positioned at the Month Entry.

As shown in Figure 3-5, the Welcome menu consists of one page of Date, Time, and User Identification questions. The purpose of these questions and the questions that will follow on the subsequent menus is to identify your CAT configuration to the Computer. In addition to the menu questions, the file name from the Configuration Statement and the applicable Name from the COM Statement are used to identify a CAT hardware configuration to the computer. When all of the menus have been filled in and the configuration is stored on a disc, the menus will be "Time-Stamped". In order to have a Time-Stamp, the current date and time is entered via the Welcome Menu. All of the above are covered in subsequent paragraphs.

You are asked to supply answers for the following questions on the Welcome Menu. The accuracy of each answer is up to you. If you mistakenly enter values which are outside the allowable range, the computer will beep, ignore your input and print an appropriate error message.

Year — Type in a four digit number, for example 1984. The program checks to see that your entry is ≥ 1984. Enter the current year.

Month — This requires one of the following three character abbreviations Jan/Feb/Mar/Apr/May/Jun/Jul/Aug/ Sep/Oct/Nov/Dec. No typing is required; use the + or − Change softkeys to select the correct month.

Day — Type in the current day of the month from 1 to 31 as a numerical entry. The program checks to see that Day is ≤ 31.

Hour — Type in the current hour from 0 to 23. This is a twenty four hour clock; PM hours are entered as 12 + the hour. The program checks to see that entries are ≤ 23.

Minutes — Enter the current minutes from 0 to 59. The program checks to see that entries are ≤ 59.

After entering the above, the program resets the computer's real time clock to the specified date and time. As long as power is left on, this date/time will be continuously updated by the computer. This date and time can be accessed with the BASIC statement "TIME$".

User ID — This question will accept any response up to 20 characters and should be used to identify the last person to change the hardware configuration and/or the application program. The User ID will appear in the header of each successive configuration menu. This is an optional entry.

When you have completed answering the questions on the Welcome Menu, press the START softkey and the next menu, the "Name Selection Menu", will appear on the screen.

Start — Pressing this key will exit the Welcome menu and cause the Name Selection menu to appear on the screen.

The Name Selection Menu

This menu, shown in Figure 3-6, consists of one or more pages listing all of the I/O function names you entered in the "COM/Names/" statement(s). You will use this menu to select a name that you wish to configure. Names are configured one at a time and you will return to this menu to configure each name. When the last name is configured, you will return to this menu to exit the configuration phase.

The heading at the top of the screen contains information which will be carried along and appear in all of the remaining configuration menus. Name and Model will be blank if you have not yet tied particular I/O functions to particular Names. The File Name in the heading is the one you assigned in the Configure ("Menu__on", "MY__CONFIG") statement. User ID is the identification you just entered into the Welcome Menu. The Time-stamp indicates "First Time" because you have not stored this configuration yet. You will see the current Date and Time (hours, minutes, seconds) in the upper right hand corner of the heading.

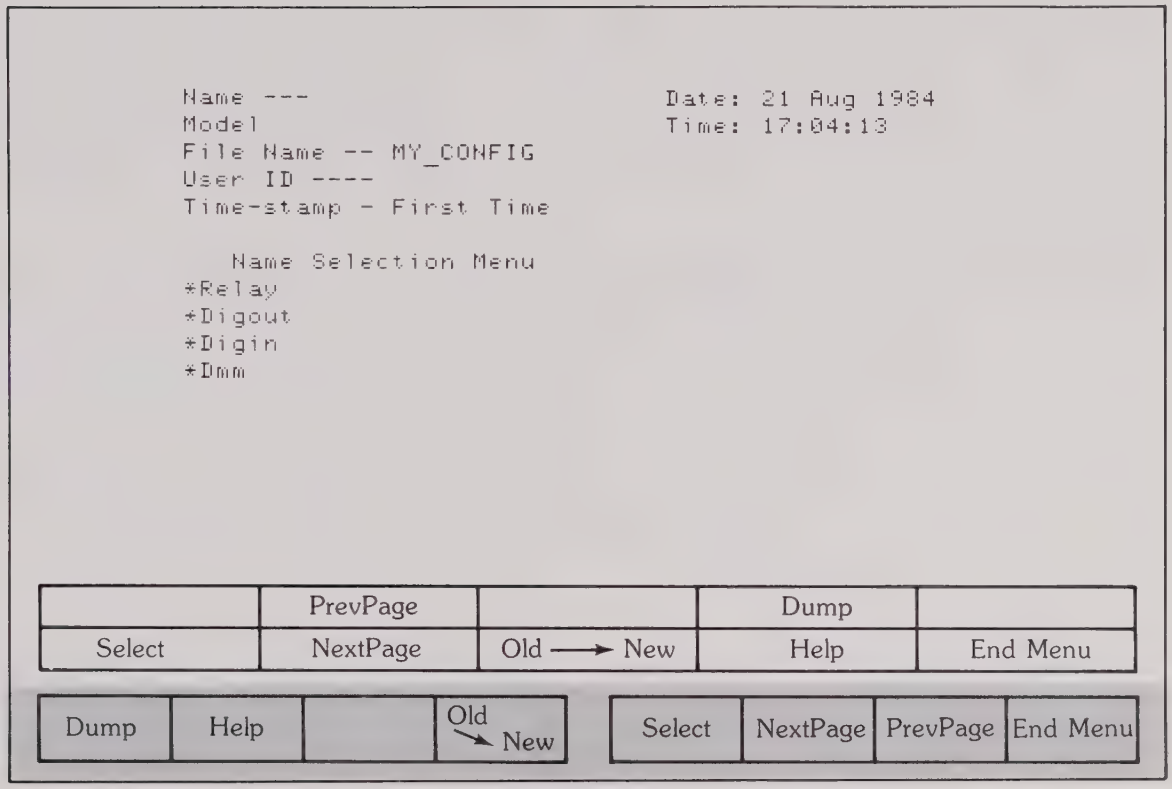


Figure 3-6. The Name Selection Menu

Throughout the configuration phase, the Time and the Date will be updated by the computer's real time clock. When the configuration phase is completed, the file is time-stamped when it is stored to the disc. The next time you access the menus using this configuration file the time-stamp on each menu will contain the date and time that the file was last stored.

Selecting a Name. The Names are listed in the order of occurrence in the "COM/Names/" statements. An asterisk in front of a name indicates that the name has not been previously configured.

Several new softkeys are part of the Name Selection Menu. These softkeys will assist you in the configuration process. The Cursor Control keys, Help softkey, and Dump softkey perform the same control functions as described previously.

Move the cursor with the Cursor Control keys to the first name that you wish to configure. There must be an asterisk (*) next to this name indicating that it has not been previously configured. To select this name press the Select softkey or the ENTER or RETURN key on the keyboard.

Select — Pressing this key will select the Name at the cursor's present location and will move you into the Function Selection Menu. Should you make a mistake here, get back to the Name Selection Menu with NextName softkey on the Function Selection Menu and reconfigure the name with the Old ⟶ New softkey.

ENTER or RETURN on keyboard — Same as Select softkey.

You may find that you want to select a name which you previously configured in order to modify some parameter. Upon reentering the Name Selection Menu, any name which you have previously selected for configuration will not have an asterisk (•). As long as the I/O function which has been previously assigned to that name is the correct model, you can press the "Select" softkey and proceed right to the specific function menu, bypassing the Function Selection Menu entirely. If the model is incorrect and you wish to specify a different one to be assigned to that name, you must first press the Old ⟶New softkey.

Old ⟶New — Unlinks the current name from its previously selected I/O function model. In order to reconfigure a previously selected name, you must position the cursor on that name, press the Old ⟶New softkey and then select the name with either the Select softkey or the ENTER key. The asterisk (•) will reappear in front of the name to indicate that it has been unconfigured.

End Menu — Use this key only after you have configured all of your named functions. Pressing the End Menu softkey causes the Exit Menu softkeys (Figure 3-11) to appear on the screen.

The remaining softkeys provide the following functions:

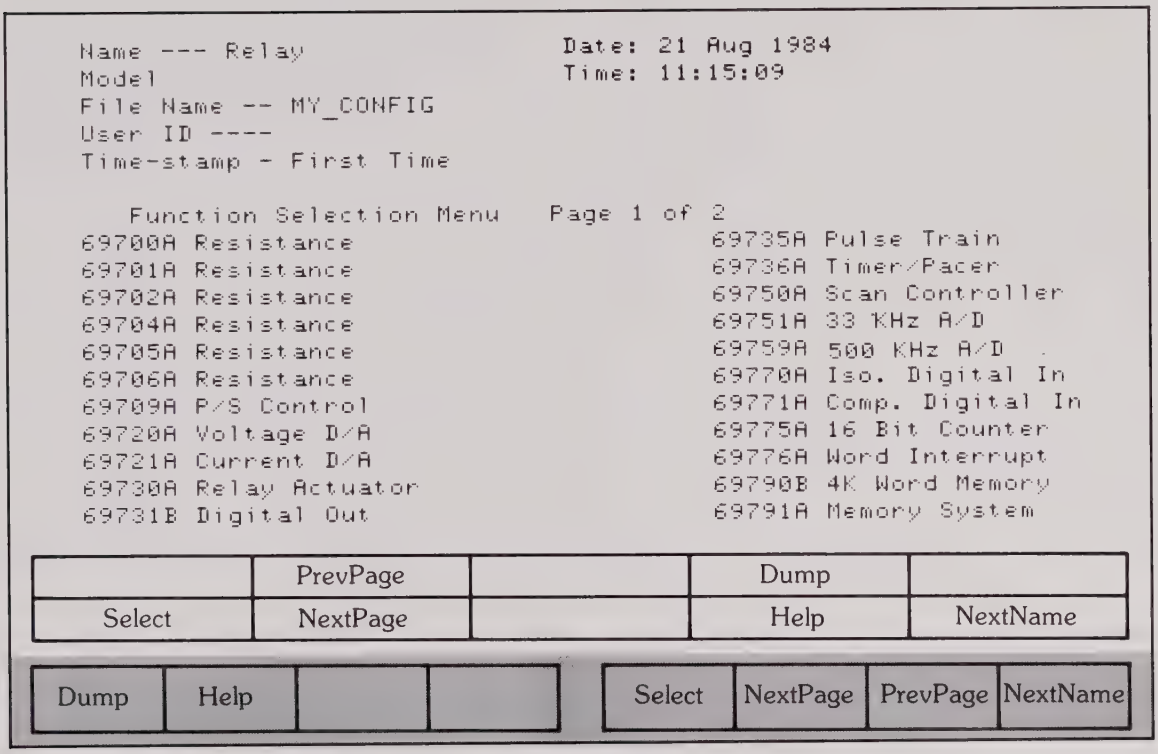
NextPage and PrevPage — Appear only if the menu has more than one page. Pressing the applicable key selects the next page or previous page of the Names listing. In an application with a large number of user Names, it may take several pages to display all of the names.

The Function Selection Menu

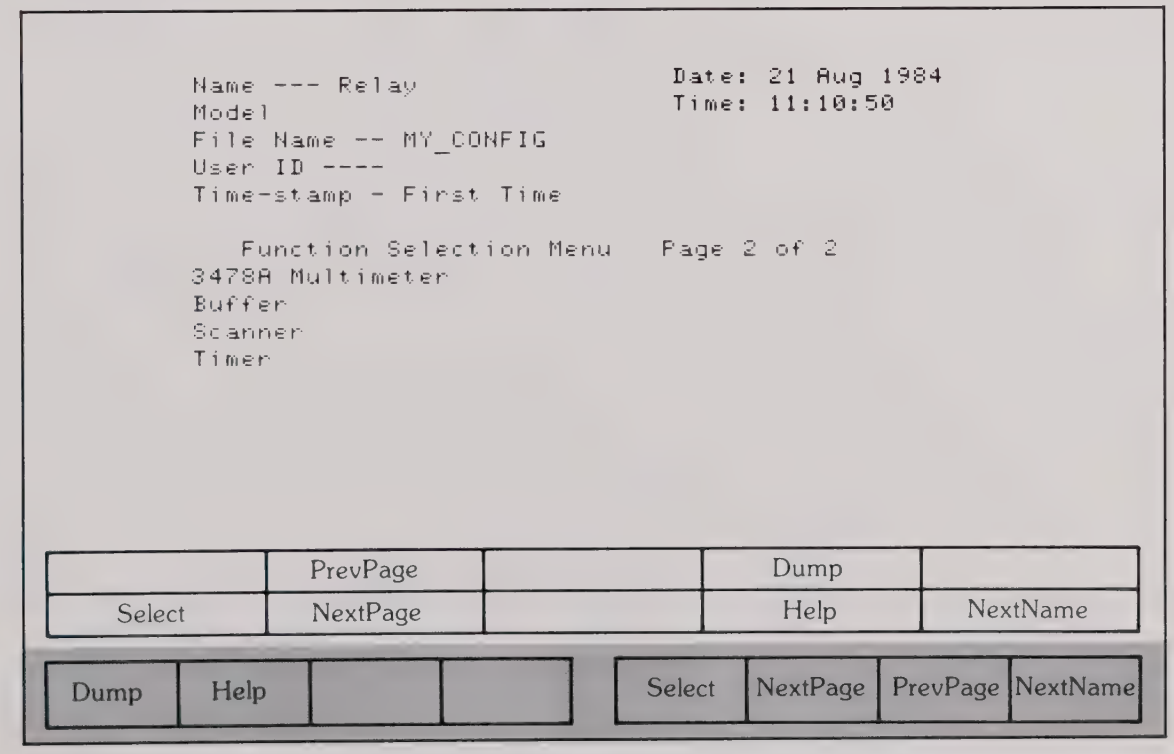
This menu, shown in Figure 3-7, consists of two pages that list the I/O functions which are supported by the 14752A CAT Programming Package. Menu and programming details for these functions are given in Chapters 5 through 9 of this manual. Table 3-1 lists each I/O function, or model, as it appears on the Function Selection Menu, the function type (Analog, Digital, etc.) and the applicable Chapter that describes the function in detail.

The purpose of the Function Selection Menu is to associate the supported functions with user specified Names. Although multiple names can be assigned to a particular function, a given name can be associated with only one function.

As shown in Figure 3-7, the heading again appears at the top of the screen; now the name selected in the last menu appears in the heading for reference. The model number is blank because you have not yet connected a function to this name. As described previously, to move from page to page, press the NextPage or the PrevPage softkeys. The function (hardware model) to be associated with the name (indicated within the heading) is selected by moving the cursor to the desired function. The Knob or the arrow keys on the keyboard can be used to move the cursor. Once the cursor is positioned, indicate your choice by pressing either the Select softkey or ENTER.



A. Page 1 of 2



B. Page 2 of 2

Figure 3-7. The Function Selection Menu

Select

Pressing this key will select the function at the cursor's present location and will move you to the Address page of the specified function menu. If you have made a mistake in selecting a function, you can correct it by pressing the Next Name softkey on the Address Page Menu which gets you back to the Name Selection Menu. Then position the cursor on the same name and toggle the asterisk back on with the Old ⟶ New softkey, and select another name.

ENTER or RETURN on keyboard

Same as Select softkey.

NextName

Use this key if you have selected the wrong name by mistake. Pressing this key will get you back to the Name Selection menu.

When you "Select" the I/O function which you wish to associate with this name, you will advance to the specified Function menu.

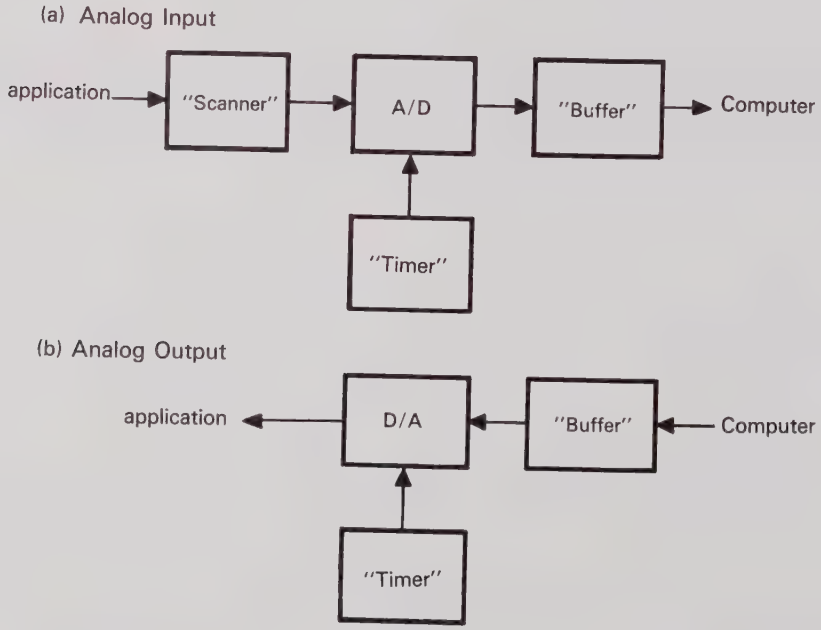


Figure 3-8. Analog Multiple Card Functions

Table 3-1. 14752A Supported I/O Functions

TYPE	I/O FUNCTIONS
Analog Input Functions (Figure 3-8a and Chapter 5)	69751A 33 kHz A/D 69759A 500 kHz A/D (−10.240 to +10.235 V, standard) (can reference Scanner or Scan Controller menu) 69750A Scan Controller (controls 69752A/754A/755A) *Scanner (can combine 69750A, 69790B/69791A, and 69752A/754A/755A) 3478A Multimeter (dc/ac volts, dc/ac amps, and 2 wire/4 wire ohms)
Analog Output Functions (Figure 3-8 b and Chapter 6)	69700A Resistance (unloaded) 69701A Resistance (8190 ohms max) 69702A Resistance (30,712.5 ohms max) 69704A Resistance (40,950 ohms max) 69705A Resistance (81,900 ohms max) 69706A Resistance (204,750 ohms max) 69709A Power Supply Controller (6012A, 6023A and 6024A Power Supplies with option 002) 69720A Voltage D/A (−10.240 to +10.235 V) 69721A Current D/A (−20.48 to +20.47 mA)
Digital Input Functions (Chapter 7)	69770A Isolated Digital Input (16 bits) 69771A Digital Input/Analog Comparator (16 bits)
Digital Output Functions (Chapter 8)	69730A Relay Output (16 contacts)—Includes a Matrix Switch Programming Application 69731B Digital Output (16 bits)—Includes a Programming Application for the 14570A AC Power Controller, Option 042
Timebase, Interrupt, and Memory Functions (Chapter 9)	69735A Pulse Train 69736A Timer/Pacer (can be used as a timebase for the Buffered A/D and Buffered D/A functions 69775A Counter (16 bits) *Timer (combines 69736A and 69775A(s) to be used as a Timebase for Buffered A/D and D/A functions) 69776A Word Interrupt 69790B 4 k Word Memory 69791A 64 k Word Memory (FIFO and Recirculate Modes) (expandable to 1 M byte with 69792A Expansion cards). *Buffer (the 69790B/69791A references either 69751A/69759A/69720A/69721A)

*multiple card functions

I/O Function Menus

You are now ready to describe the exact configuration of a named I/O function to the computer. Describing the configuration consists of supplying the address of the I/O function, selecting data conversion and limit checking options, and specifying the initial conditions that the function will assume when an "Init" or "System_init" statement is executed. There can be as many as four pages to a Function Menu; each set has been tailored according to the function it defines. The following discussion explains how to answer the menu's questions using the 69730A Relay Output card's menu as a example. Should you have any questions about a response necessary to complete a specific function's configuration, refer to the appropriate function's description given in Chapters 5 through 9 of this manual.

The Address Page

The first page of the function specific menu is the address page as shown in Figure 3-9. The page is similar for all I/O function menus. At the top of the screen, you will see the heading information with the name and the function model number filled in. This is page 1 of 2 as indicated. When a particular page is completed, use the NextPage softkey to advance to the next page. The PrevPage softkey can be used to get back to the previous page.

In the center of the screen you will see a number of questions relating to the I/O function's address in the system. The 14752A product provides default values for most of the address parameters. The one exception is Slot; you **MUST** respond to the Slot address question as the computer has no way of knowing into which Multiprogrammer slot you have inserted a particular I/O card. Should you try to enter an illegal response into any of the address parameters, the computer will beep and an error message will appear on your screen immediately below the Timeout parameter.

Figure 3-9 shows the questions and default values that appear on the 69730A's Address Page. A particular question (address parameter) is chosen by positioning the cursor on it. The Help softkey is available to aid you if you forget how to use the other keys. To input an address parameter, position the cursor on the particular question, enter the numeric value from either the main keyboard or the numeric keypad, and then press the Enter softkey (or ENTER or RETURN on the keyboard). Note that advancing the cursor to the next question Knob or arrow keys will also "enter" your response (or Default value).

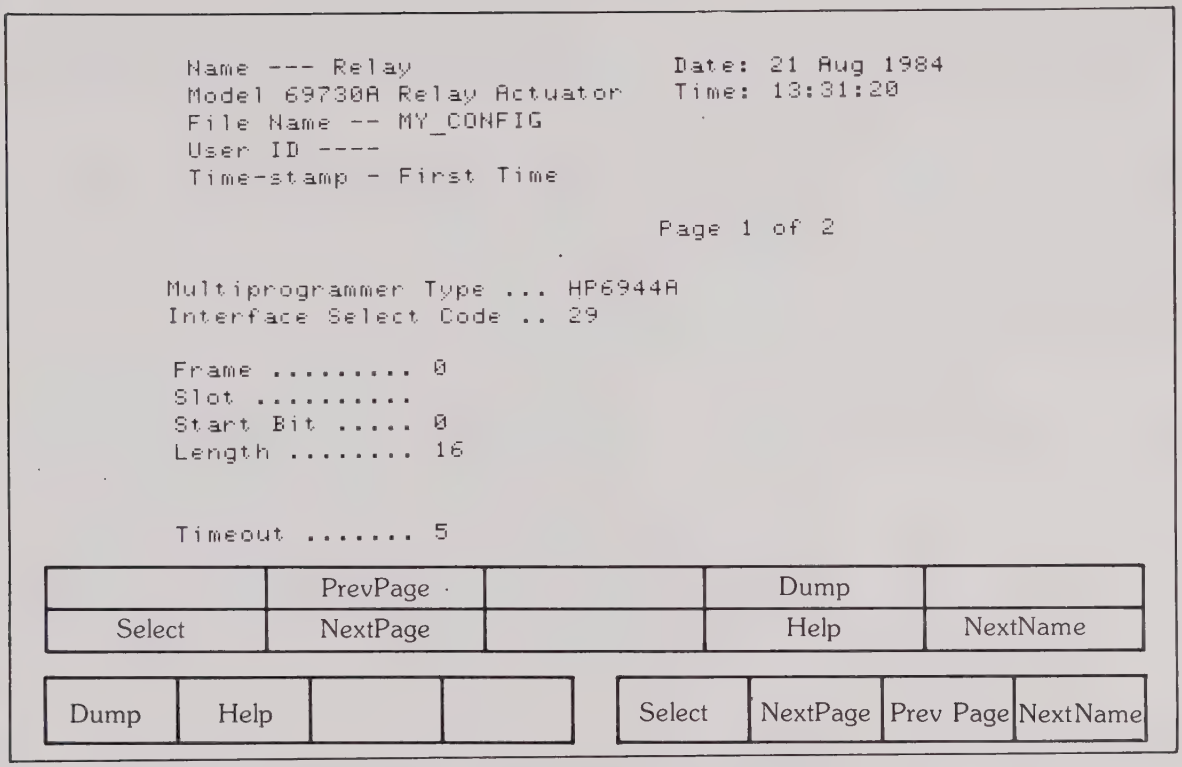


Figure 3-9. 69730A Relay Actuator Function, Menu Page 1 of 2 (Address Page)

Enter

Pressing this key will load the value you have input from the keyboard; the cursor will automatically move down to the next question on the page. Should you make a mistake, press the Clear softkey and enter your response correctly.

ENTER or RETURN on keyboard

Same as Enter softkey.

Clear

Pressing this key clears the keyboard entry where the cursor is positioned. Use this key to clear incorrect entries.

Change – and **Change +**

Used to select from a set of predefined responses to alphanumeric questions (e.g. Yes/No). Appear in place of the Clear softkey when a predefined response is required (see Data Conversion page, Menu Page 2).

NextName

Pressing this key returns you to the Name Selection Menu (see Figure 3-6). Use this key after you have answered every question on every page. Once you return to the Name Selection Menu, you can select another name to configure or you can exit the configuration phase by pressing the End Menu softkey on the Name Selection Menu.

Now take a few minutes to fill in the address parameters on page 1 of this menu (see Figure 3-9). When you have completed all the responses, press the NextPage softkey to advance to the Data Conversion information on page 2 of 2.

Multiprogrammer Type

This is the model number of the Multiprogrammer to be configured. Use the change + or − softkeys to select 6942A or 6944A Multiprogrammer. The default is 6944A and is selected by positioning the cursor down to the next parameter.

Interface Select Code

Appears only when configuring the 6944A Multiprogrammer. This is the select code of the 98633A Multiprogrammer Interface card that is used by the 14752A. This card plugs into the backplane of the computer. It is designed to optimize instruction and data transfers between the computer and the 6944A Multiprogrammer. The 98633A has a legal range from 8 to 31 and has been preset in the factory to "29". The 14752A defaults the Multiprogrammer Interface to 29 on this menu. To accept this default, position the cursor down to the next parameter. If you need to verify the address switch setting to the 98633A, make sure the power is turned off at the computer before removing the interface card.

HP-IB Select Code
HP-IB Address

Appears only when the 3478A Digital Multimeter or 6942A/6943A Multiprogrammer menu is being configured. The Select Code is the address of the 98624A HP-IB Interface card and the Address is of the particular 3478A Digital Multimeter on the HP-IB bus. The 98624A Interface card has a range from 8 to 31 and has been preset in the factory to "8". The 14752A defaults the HP-IB Select Code to 8 and the Address to 22 on this menu. To select the default values, position the cursor down to the next parameter. If you need to verify the address switch setting on the 98624A, make sure the power is turned off at the computer before removing the card. If you need to check the address setting of the actual instruments, visually verify the HP-IB bus address switches (refer to Chapter 2 of this manual).

The following Frame and Slot parameters will appear on the Address menu page of each I/O card.

Frame

This parameter specifies the frame address of the Multiprogrammer which holds the specific function I/O card which you are currently configuring. It has a legal range from 0 to 7. The default value is 0. To select the default value, position the cursor to the next parameter on the menu. For details on setting the Frame address, refer to Chapter 2 of this manual.

Slot

This parameter identifies the location of the function I/O card within the Multiprogrammer. This parameter **MUST** be entered in order to proceed into the configuration process. It has a legal range of 0 to 15 with no default value. If you should forget to enter the Slot number and try to advance to the next parameter, the computer will beep and an error message will appear at the bottom of your screen. To recover you must keystroke in your response, and then advance to the next parameter. If you move the I/O card corresponding to this named function into another slot, you must come back to this menu and update the slot address.

The next two parameters, Start and Length, are included only on the Address page of the menus for the 16-bit 69730A Relay Output, 69731B Digital Output, 69770A Isolated Digital Input, and the 69771A Digital Input/Analog Comparator functions.

Start and Length

These parameters specify which bit or group of contiguous bits on the card are assigned to the user's names. The 16-bits on each card are designated bit 0 (LSB) through bit 15 (MSB). The Start parameter specifies the first bit in a group; it has a default value of 0 (or bit 0). The total number of contiguous bits in the group is specified by the Length parameter; it has a default value of 16 (specifies all 16 bits, bits 0 through 15). The software will check to insure that Start plus Length are $\leq$ 16. Refer to Chapters 7 and 8 for specific examples on how to use the Start and Length parameters.

Timeout

This parameter specifies the length of time in seconds that the computer will wait for a name to complete its main I/O function. Timeout is normally used in conjunction with the "Output", "Input", and "Wait_for" statements. The Timeout paramenter default is set at 5 seconds. Should you want to program a different timeout (e.g., 15 milliseconds), then enter .015 into this menu. When a function takes longer to complete its operation than the Timeout value allocated for that function, the program will be halted and an error message will be generated. The legal range for Timeout is from 0.001 to 128,000 seconds. These values can be entered in the menu in decimal or in scientific format as follows:

0.001 or 1E-3
128,000 or 1.28E5

The Timeout feature is provided as a way of notifying you if your I/O system is hung up. You should always specify a timeout value that is greater than the time it will take the I/O function to complete its operation.

The Data Conversion Page

Pressing the NextPage softkey will bring you from the Address Page (page 1) to page 2. On all menus except the 69750A Scan Controller, Scanner and Buffer, page 2 is the Data Conversion page which gives you a choice of the units you use to program the function. This choice allows you to program a function in its "Standard" units (e.g. a 69720A D/A is programmed in volts) or in units that you specify which make sense for your particular application. For example, assume that the 69720A D/A is connected to a transducer to control the flow through a valve in gallons/minute. By entering the appropriate values on the 69720A's Data Conversion page to scale the units, you can program the function in user defined units (gallons/minute) instead of the function's standard units (volts).

The numeric responses require a REAL number entry. For Series 200 and 300 BASIC, the valid range for REAL numbers is:

$$-1.797\ 693\ 134\ 862\ 315 \times 10^{308} \text{ thru } 1.797\ 693\ 134\ 862\ 315 \times 10^{308}$$

The smallest non-zero REAL value is approximately:

$$\pm 2.225\ 073\ 858\ 507\ 202 \times 10^{-308}$$

A REAL number can also have the value zero.

To enter a REAL number response, position the cursor next to the parameter of interest, type in the number with or without scientific notation (e.g. 1.25E-3, or .00125), then press the Enter softkey.

Should you try to enter a value that is out of range, the Computer will beep and an error message will appear at the bottom of the screen.

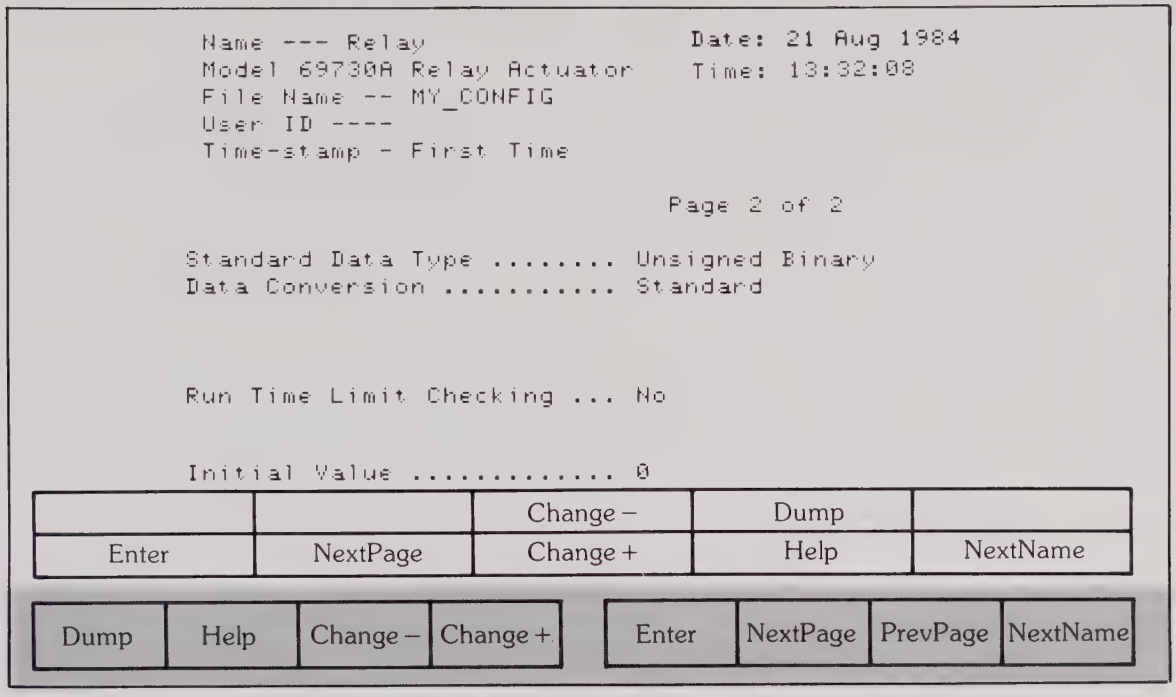
Now take a few minutes to fill in the parameters on the Data Conversion page (see Figure 3-10). When you have completed filling out this page, press the Next Name softkey to return to the Name Selection menu. When the Name Selection menu appears, you can select another name to configure or you can exit the configuration phase (see Figure 3-3).

Standard Data Type

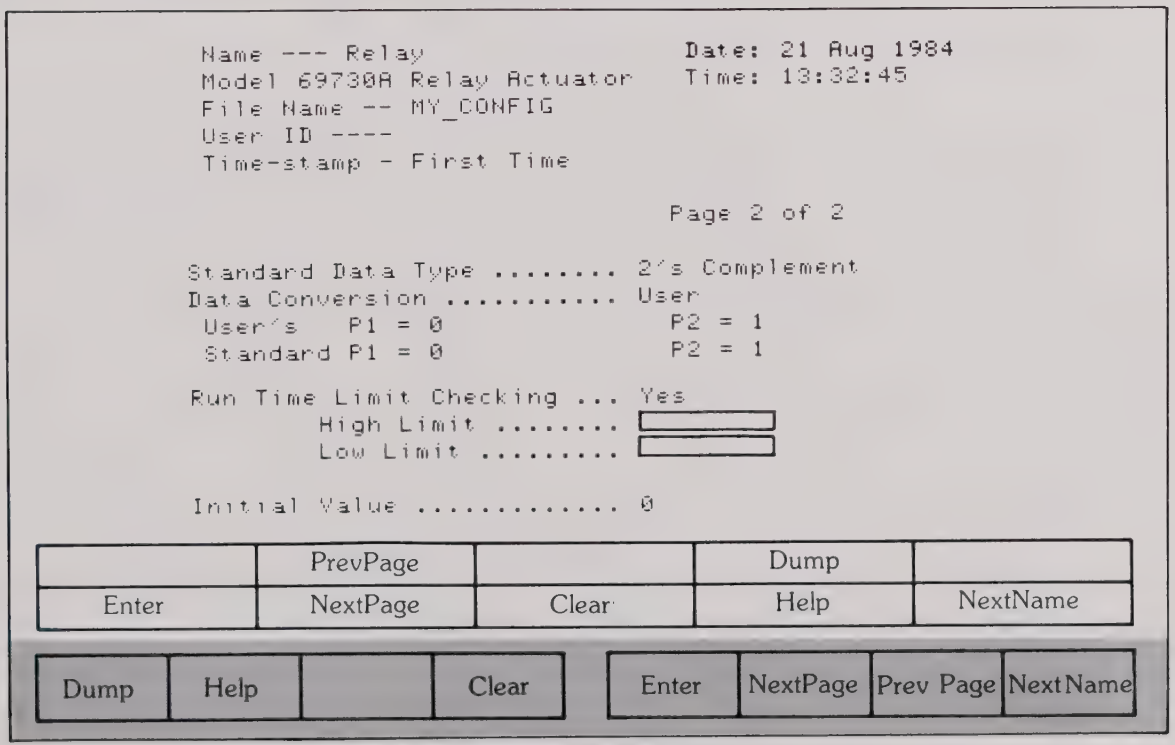
This parameter applies to all Digital Input/Output functions and specifies the data type for the function. It has two predefined responses: "Unsigned Binary" (default) or "2's Complement". Use the + and—Change softkeys to select your response. For Unsigned Binary, you can program in positive numbers only; for 2's complement, you can program in positive or negative numbers. Refer to the applicable function's description in Chapter 7,8, or 9 for the range of numbers permitted for the particular function.

Data Conversion

Specifies how the data, which is sent to an Output function or returned from an Input function is to be scaled by the 14752A. The menus for the Digital Input/Output and Timebase functions have two predefined responses: "Standard" or "User" conversion. The menus for the Analog Input/ Output functions have three predefined parameters: "Standard" conversion, "User" conversion, or "None". Use one of the Change—or + softkeys to select your response. Each type of conversion is described as follows:



A. Default Parameters



B. Optional Parameters

Figure 3-10. 69730A Relay Actuator Function, Menu Page 2 of 2 (Data Conversion Page)

1. Data Conversion for Digital Input/Output Functions

   a. "Standard" conversion means that the data values sent to an output function or received from an input function are numbers in the form of Unsigned Binary or 2's Complement values, whichever was selected as the Standard Data Type. Standard is the default Data Conversion for these functions. For Unsigned Binary, the range of numbers for a 16-bit function is from 0 (all bits off) to 65535 (all bits on). For 2's Complement, the range of numbers for a 16-bit function is from −32768 (all bits off except bit 15) to +32767 (all bits on except bit 15). Refer to Chapters 7,8,9 for "Standard" conversion details related to specific digital functions.

   b. "User" conversion allows data values to be specified in terms of the user's application rather than in terms of the function's Standard units (numbers described above). As shown in Figure 3-10, four additional entries (User's P1 and P2, Standard P1 and P2) appear on the menu when "User" conversion is selected. These entries allow you to fill in the ratio of user defined units and the function's standard units so that the proper conversion is made. The P1 entries default to 0; the P2 entries default to 1. A "User" conversion example is given below in the discussion for Analog Input/Output functions. Refer to Chapters 7,8,9 for "User" conversion details related to specific digital functions.

2. Data Conversion for Analog Input/Output Functions

   a. "Standard" conversion means that the data values sent to an output function or received from an input function represent the standard units (volts, milliamps, ohms, etc.) for that particular function. A few examples are shown below. Refer to Chapters 5 and 6 for "Standard" conversion details related to specific analog functions.

Function	**Standard Units**	**LSB**	**Program Example**
69701A Resistance	0 to 8190 Ohms	2 ohms	Output (Resistance,400)
69720A Voltage D/A	−10.240 to +10.235 Volts	.005 V	Output(Volts,5.250)
69721A Current D/A	−.02048 to +.02047 Amps	.00001A	Output(Amps, −.01745)
69751A A/D 69759A A/D	−10.240 to +10.235 Volts	.005 V	Input(Analog,Value)

   b. "User" conversion allows data to be specified in terms of the application rather than in terms of the function's standard units. As shown in Figure 3-10, four additional entries (User's P1 and P2, Standard P1 and P2) appear on the menu when "User" conversion is selected. These entries allow you to fill in the ratio of the user defined units and the function's standard units so that the proper conversion is made. User conversions assume a **LINEAR** relationship between the User and the Standard units. A conversion example is given below. Refer to Chapters 5 and 6 for specific analog function examples.

c. "None" means that the data values sent to an output function or returned from an input function represent the number of LSB's for that particular function. This is the fastest way of programming an analog input or an analog output function and consists of sending to an output function or receiving from an input function, the decimal number of LSB's required. For example, to program the 69701A Resistance Output function, sending the decimal value 4095 would program the output resistance to 8190 ohms; (4095 X LSB value or 4095 X 2 ohms). Refer to Chapters 5 and 6 for conversion details related to specific analog functions.

User Conversion Example

Assume that the 69751A A/D Converter card, which has a -10.240 to +10.235V input range, is connected to a transducer that converts a magnitude in pounds into a voltage reading. The transducer has an output range from 1 to 10 volts that corresponds linearly to a magnitude in pounds from 30 to 250 pounds. To allow the 69751A A/D to return readings directly in pounds instead of in volts, select User conversion and enter the ratio of pounds and volts as follows:

```
Data Conversion.....User
  User's      P1. 30      P2. 250
  Standard    P1.1        P2. 10
```

Additional examples of User conversion are given in Chapter 5 through 9.

Run Time Limit Checking This parameter appears only on the menus of output functions. It has two predefined responses: "Yes" or "No". Select your response with the + and Change softkeys. The default is "No". If "Yes" is selected, the value sent to an output function will be checked against the high and low limits that you specify. There is additional software overhead time to program the function when limits are checked.

High/Low Limit These entries appear on the menu only when Run Time Limit Checking is selected. They specify the upper and lower boundaries for the output data. Type in any REAL number in the valid range. Your choice of limits must be **Consistent with the Data Conversion Option that you have selected.** With Run Time Limit Checking selected, if your program mistakenly tries to output a value which is not within these limits, an error (802) will be generated and your program will halt. These parameters **MUST** be entered; there are no default values.

Initial Value This parameter appears on the menus for output functions only. It gives you the ability to preset an output function with an initial value. Either an Init(Name) or a System_init statement will cause the output function (except for the 69735A Pulse Output and the 69736A Timer/Pacer functions) to go to this value. You must enter a REAL number that is **Consistent with the Data Conversion Option that you have selected.** The default values for this parameter depend upon the particular output function.

Additional Function Menu Pages

The more complex functions have more than two pages that specify more parameters that you may select. They are described in detail by function in Chapters 5 through 9.

Storing the Configuration File

When you have completed configuring all of your names(or as many of them as you wish to do in this session), prepare to store the configuration information on a disc and exit the configuration phase by pressing the End Menu softkey on the Name Selection Menu (see Figure 3-6). The Exit Menu softkeys will then appear on the computer screen as shown in Figure 3-11.

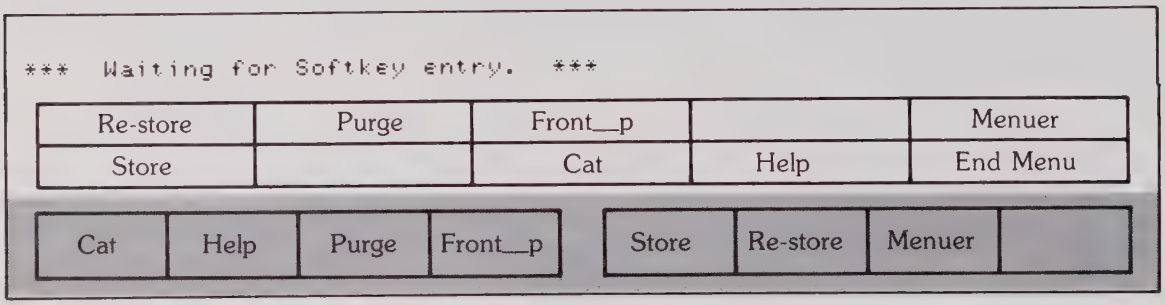


Figure 3-11. Exit Menu Softkeys

The Exit Menu softkeys allow you to store the configuration information you have just finished specifying and then exit the configuration phase. After storing this information on a disc, you have the choice of running your application program, calling the "Soft Front Panel" utility program, or returning to the configuration phase at the beginning. Figure 3-12 summarizes the three classes of capabilities provided by the Exit Menu softkeys; each softkey is also covered by a separate discussion in this section. Figure 3-13 illustrates the effect of the softkeys on program flow.

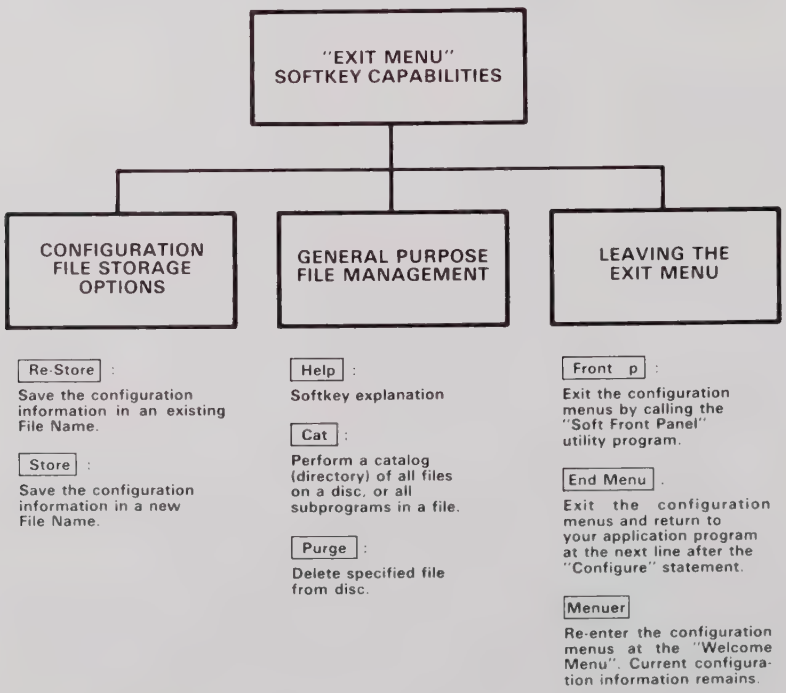

"EXIT MENU" SOFTKEY CAPABILITIES
CONFIGURATION FILE STORAGE OPTIONS
GENERAL PURPOSE FILE MANAGEMENT
LEAVING THE EXIT MENU
Re-Store :
Save the configuration information in an existing File Name.
Store :
Save the configuration information in a new File Name.
Help :
Softkey explanation
Cat :
Perform a catalog (directory) of all files on a disc, or all subprograms in a file.
Purge :
Delete specified file from disc.
Front p :
Exit the configuration menus by calling the "Soft Front Panel" utility program.
End Menu :
Exit the configuration menus and return to your application program at the next line after the "Configure" statement.
Menuer :
Re-enter the configuration menus at the "Welcome Menu". Current configuration information remains.

Figure 3-12. Capabilities Provided by the "Exit Menu" Softkeys

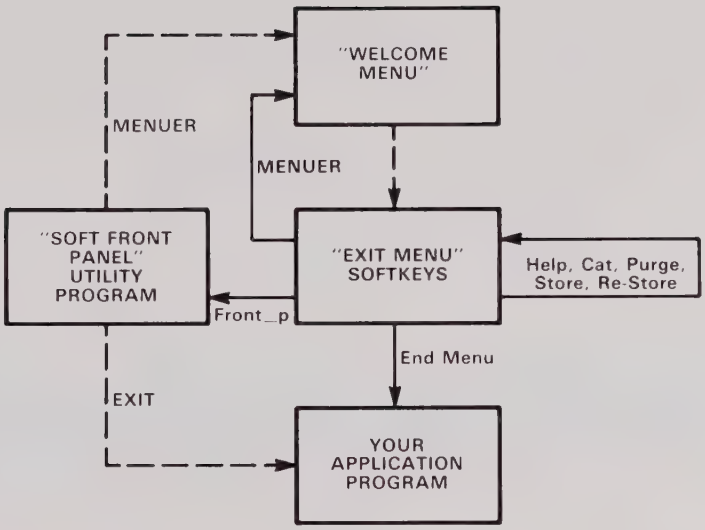


Figure 3-13. Program Flow for "Exit Menu" Softkeys

Configuration File Storage Options

The configuration information can be stored in a type BDAT file on a disc using the softkeys explained in this section. There is no other way, and no other time during the execution of your application program, that you can save this information. You are not required to store the information; you can exit the configuration phase without doing so, and proceed with the execution of your program. However, it is strongly urged that you do store it, because the next time you RUN your program any previous configuration information will be lost unless it has been stored on a disc.

There are two softkeys used for storing your configuration on a disc, depending upon whether or not the file specified in your "Configure" statement already exists.

Re-Store — Used to store the configuration information in a file that already exists. Pressing this key has the effect of first PURGEing the existing file and then STOREing the new one. For example, if you wanted to store your configuration information in the BDAT file "MY_CONFIG"on the 14752A CAT Programming Package disc, you would use this softkey, since "MY_CONFIG" already exists.

Store — Used to store the configuration information in a file which does not exist. For example, if you wanted to store the configuration information in the BDAT file "NEW_FILE", you would use this softkey, because "NEW_FILE" is a new file name (has not been previously stored on the disc). Remember that to access this new file name, you will have to edit the Configure Statement in your program.

When either of the above softkeys is pressed, the File Specifier Screen appears on the screen. This screen is used to specify the file name in which the configuration information will be stored.

The File Specifier Screen

This screen, shown in Figure 3-14, allows you to tell the system under what name, and on which disc drive, your configuration file is to be stored.

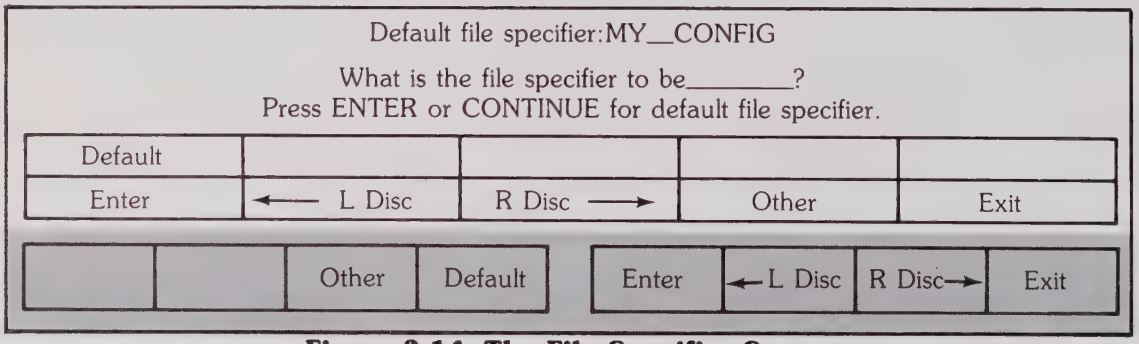


Figure 3-14. The File Specifier Screen

The "Default File Specifier" shown will always be that of the file contained in the Configure statement in your program. In most cases, you will want to use the default file specifier as the destination for your configuration information. If this is the case, merely press the ENTER or CONTINUE key. The file will be written to the disc, and the Exit Menu softkeys will appear on the screen when the storage operation is completed. If this is what you have chosen to do, you can now leave the the configuration phase and resume execution of your application program at the line following the Configure statement.

Specifying a Different Destination File

The file specifier screen provides several other options for the storage of your configuration file. These options allow you flexibility but without complicating the most common way (described above) that configuration storage is done.

When any of the following softkeys are pressed, a new line will appear on the bottom of the display. This is where the new file name and/or mass storage device are specified. Use the knob or the arrow keys to move to the first character that is to be changed, and begin overtyping. Once you have corrected the entry, press ENTER or CONTINUE to proceed.

Default — This softkey is used when the configuration file is to be stored on the current system mass storage device. Only the file name appears on the new line; it can be changed as described above if desired.

Internal — This softkey appears only on 9826 based systems. It is used when the configuration file is to be stored on the internal disc drive of the 9826. Although both the file name and the disc drive specifier appear on the new line, only the file name can be edited as described above.

←— L Disc — This softkey appears wherever the system has multiple disc drives. It is used when the configuration file is to be stored on the left drive.

R Disc ⟶ This softkey appears wherever the system has multiple disc drives. It is analogous to the L Drive softkey above, but refers to right disc drive.

Other This softkey is used when the configuration file is to be stored on a mass storage device supported by BASIC but not covered by the other softkeys. For this case, up to 26 characters are provided for you to edit and enter the entire file specifier, name and device, as described in the BASIC Language Reference Manual.

Exit If you have made an incorrect entry to get to the File Specifier Screen, this key will return you to the Exit Menu softkeys.

File-Related Errors

Several errors related to mass storage operations can occur during the Exit Menu phase. The major ones, their effect, and your corrective actions are listed below:

"File not found"

- You tried to "Re-Store" a file that did not exist. Use "Store" instead.

"File exists. Use "Re-Store" to write over the old file."

- You tried to "Store" a file that already exists.

"Drive door open, or media not installed."

- This message can occur for many of the softkeys. Close the drive door or install the media, and re-execute your command.

"Mass storage media overflow."

- This message ocurs when you try to "Store" or "Re-Store" a configuration file and there is not enough room on the disc for it. As indicated in the text of the message, you have three options:

1. You can purge any old or unused files on the disc. This is done by using the Purge softkey as described later in the "General File Management Section".

2. You can insert another disc in the drive and try the command again. If the disc you choose has not been initialized, the system will inform you of this and allow you to have it initialized at this time.

3. You can store the file on another mass storage device, if one is available.

You will notice that the Exit Menu softkeys have reappeared; take the corrective action of your choice and re-execute your command.

"The disc has not been initialized.
Do you want it to be initialized?"

- If the disc you have tried to Store or Re-Store the configuration information on has not been initialized, this message will appear. Two softkeys appear with "Yes" and "No" choices. If you answer "No", you will return to the Exit Menu softkeys. If you answer "Yes", the initialization process will begin; the bottom line of the display will give you feedback that initialization is occurring. Initialization can take several minutes to complete. After the disc is initialized, the "Store" or "Re'Store" operation that you were doing will proceed automatically. When it is concluded, the Exit Menu softkeys will reappear.

General Purpose File Management

There are two softkeys in the Exit Menu which can assist you in storing or re-storing your configuration information if file-related errors occur.

Cat — Gives you either a list of all file names on a specified disc or a list of all subprograms in a specified file. Pressing the softkey gets you to the File Specifier screen, described previously. If you specify only a disc drive specifier and omit the file name, you will get a catalog of all the files on the disc. If you specify a filename in addition to the drive specifier, you will get a list of the subprograms contained in that file.

Purge — This key can be helpful when a "Mass storage media overflow" error occurs. You can use it to purge old or unused files on your disc in order to make room for the new file containing your configuration information. Pressing the softkey gets you to the File Specifier Screen, described previously.

The Help and the Dump keys provide the same functions as previously described for the configuration menus.

Leaving the Exit Menu

There are three actions you can take to leave the Exit Menu softkeys. Each is described below. After you have read over all three options, select the one which you wish to perform at this time.

Menuer — Pressing this key will return you to the beginning of the configuration phase; the "Welcome Menu" described in the beginning of this chapter. The configuration information that you have specified up to this point will remain in the computer's memory, even if you have not yet stored it on the disc.

You would use this softkey to return to the configuration menus whenever you come to the Exit Menu in the middle of a configuration session and wish to resume it.

Front__p

Pressing this key will execute the "Soft Front Panel" utility program. You must insert the 14752A Utility Programs disc in the drive prior to pressing this key. If you do not insert the disc, a "File Not Found" error screen, similar to Figure 3-2, will appear. Insert the proper disc in the drive and press ENTER or CONTINUE.

The "Soft Front Panel" utility program allows you to interactively select and execute 14752A program statements from the keyboard. If you wish to run the Soft Front Panel at this time, press the Front__p softkey and turn to Chapter 4, where the operation of this program is described in detail. It is recommended that you store your configuration information before pressing the Front__p softkey and executing the utility program. If the latest configuration information has not been stored, the message

"Do you wish to continue without storing?"

will appear along with "Yes" and "No" softkeys for your answer. Although you can get back to the configuration menus from the Soft Front Panel, the configuration information will be lost unless you have stored it.

End Menu

Pressing this key will end the configuration phase and resume execution of your application program at the line following the Configure statement. As stated in the Front__p discussion above, you should always store the configuration before continuing. If the latest information has not been stored, and you press the End Menu key, the message

"Do you want to continue without storing"

will appear along with "Yes" and "No" softkeys for your answer. Unless you have stored the configuration information on the disc, it will be lost the next time that you RUN your application program.

Once you press the End Menu key, and after a short delay, the message "START OF PROGRAM" (line 830 of the EXAMPLE application program) will be printed on the screen, followed by the message "END OF PROGRAM" (line 870 of the EXAMPLE application program). Any statements which you have inserted between these two lines would, of course, also be executed in the proper order.

Both the End Menu and the Front__p softkeys cause the configuration phase to be concluded. This is the first point in the execution of your program where the 14752A software talks to the hardware in your system by using the information you have supplied in the menus. The software makes the following checks before you can RUN your application program or use the "Soft Front Panel" to operate your functions.

The software first determines whether or not the 98633A Multiprogrammer Interface or 98624A HP-IB Interface is present at the select code specified in the menus. If not, the following message appears:

```
14752A: Interface card Not Present
        detected by Init_hardware
        at Select Code n
```

Turn off power to the computer and install the interface card if it is missing or check the switch settings as described in Chapter 2 if the card is present.

The software next checks to see that a 6944A Multiprogrammer exists at the Select code and Frame specified.

```
14752A: Multiprogrammer Not on Line
        detected by Init_hardware
        at Select Code 29
           Frame       0
```

If you get the error message you should check that the address switches of the 6944A are set to the correct frame address (see Chapter 2 of this manual).

The software then checks that the proper I/O card is in the frame and slot where the I/O function menu says it should be. If the wrong card or no card, is found in the slot for that function name, the error message

```
14752A: Specified Card Not in Slot
        detected by Init hardware
        for model XXXXX
        The I/O Function is associated with the name "XXXXX"
```

will be generated. You should insure that the proper I/O card is inserted in the specified frame and slot for the function name indicated. If you have a 3478A Multimeter in your system, you may get the error message, "14752A: HP-IB Bus Timeout detected by Init hardware", if the device is not at the HP-IB address specified. You should check that the 3478A is turned on, and that the address switches of the 3478A are set to the correct address (see Chapter 2). Also, if you have more than one function name which references the same 3478A, you should insure that they all specify the correct address.

Programming with the 14752A

Application programs are written using a combination of 14752 CAT Programming Package statements and BASIC statements with the I/O function names and data conversion options (scale factors) that you have defined during the configuration phase.

The various BASIC statements are discussed in detail in the BASIC Language Reference Manual.

The programming statements for each function of the 14752A CAT Programming Package are divided into three types: Fundamental Statements, Advanced Statements, and System Statements. Since each of the 14752A statements is a call to a subprogram rather than a keyword of the BASIC language, you should type them in mixed mode, that is, with the first letter capitalized and all other characters lower case.

Fundamental Programming Statements

These statements are the only ones which need to be used to cause the I/O function to perform its main operation. They generally utilize the operational parameters previously specified during the configuration phase. Most of these statements are generic; that is, they are identical for many different functions.

Advanced Programming Statements

These statements allow the full range of performance and versatility of the I/O function to be utilized. Typical uses are to control interrupts from I/O functions, change the operating mode of multi-mode functions, and to modify the parameters specified during the configuration phase. These statements are often unique to a given function. Use of these statements requires detailed knowledge of the Multiprogrammer system and the I/O functions. The use of these statements is described in Chapter 10, Advanced Programming.

A complete summary of all the 14752A statements and syntax is given in Chapter 11. Programming examples as well as the advanced programming statements that can be used with each I/O function are described in the applicable chapter (See Table 3-1).

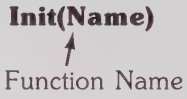
System Statements

These statements perform auxiliary functions and system-wide operations within the 14752A CAT Programming Package (See Chapter 4).

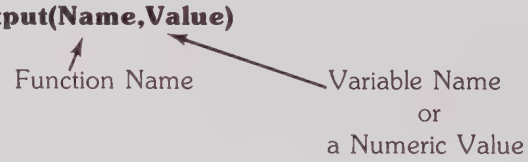
Statement Syntax

The three fundamental statements discussed in this Chapter have the following syntax.

1. 14752A Statement: (All Functions)



2. 14752A Statement: (Output Functions)



3. 14752A Statement: (Input Functions)

Function Name

The "Name" used in each 14752A statement must be one of those listed in the COM /Names/ statement. It can be up to 15 characters long and it can include the underbar (_) character. Names may be typed using any combination of upper case and lower case letters. It is good practice to type the first letter upper case and the remaining letters lower case.

Variable Name

Variable names, like I/O function names, should be chosen to enhance the readability and documentation of your program. Any name up to 15 characters long as defined above can be used. The variable name "Value" is used in the examples above.

In Output statements: "Value" can be any REAL number as well as a REAL variable, or REAL array element.

In Input statements: "Value" can only be a REAL variable or REAL array element. Input statements are used to input a REAL number and store it into "Value".

Fundamental Statement Descriptions

Init(Name)

This statement is used to initialize the named I/O function. Each function **MUST** be initialized before use; all can be done at once with a "System_init" statement. Init disables any interrupts from the function and sets the data conversion and limit checking options specified in the function's menu. The Init statement will program initial output values at an output function's edge connector (except the 69735A Pulse Train and 69736A Timer/Pacer). In control applications, you should select an initial value that represents the "safe" state of such outputs.

Output(Name,Value)

This statement is used to program the named output function to the given "Value". The Output statement will program the function in units according to the data conversion option selected in the menu for the named function. The value programmed will appear at the function's edge connector (except the 69775A Counter and 69776 Interrupt functions). If you have selected Run Time Limit Checking on the named output function's menu, Output will first check the programmed value against the high and low limits set in the menu before sending the value to the function.

Input(Name,Value)

This statement is used to input the value at the edge connector of the named input function and store it into the variable "Value". The number which is returned in "Value" will be in units determined by the data conversion option selected in the menu for the named function.

The 14752A CAT Programming Package includes a set of over 160 program statements. The three statements described above are among the most commonly used statements. Syntax requirements for each of the 14752A statements are given in Chapter 11. Programming examples for each I/O function are given in the applicable Chapter (5 through 9, see Table 3-1). These examples will help you in writing your application program(s). Read Chapter 4 for information on program development and handling run time errors. Advanced programming concepts including interrupt handling are covered in Chapter 10.

Entering 14752A Statements into the EXAMPLE Application Program

You can insert your own program lines into the EXAMPLE program (see Figure 3-1) by entering the EDIT mode and positioning the cursor to line 840. Then, press INS LN, or INSERT LINE and begin to keystroke your program. At the end of each line, press the ENTER or RETURN key. In the example below a short program for the 69730A Relay Actuator (in slot address 11) is inserted into the EXAMPLE program. Assume that the following parameters were selected on Menu pages 1 and 2 during the configuration phase:

Menu Page 1 of 2	Menu Page 2 of 2
Name = Relay Model = 69730A Relay Actuator Frame = 0 Slot = 11 Start Bit = 0 Length = 16 Timeout = 5	Standard Data Type = Unsigned Binary Data Conversion = Standard Run Time Checking = No Initial Value = 0

Example Program Insert

```
810 ! ----------BODY OF APPLICATION PROGRAM----------
820 !
830 PRINT "START OF PROGRAM"
840 ! Relay Actuator Example Program
841 !
842 ! Set up the variables to close the odd
843 ! numbered relay contacts and then
844 ! open the odd contacts and
845 ! close the even numbered contacts
846 !
847 Odd_contacts=43690
848 Even_contacts=21845
849 !
850 ! close the odd contacts
851 !
852 Output(Relay,Odd_contacts)
853 !
854 ! Wait 3 seconds and then open
855 ! the odd numbered contacts and
856 ! close the even numbered contacts
857 !
858 WAIT 3
859 Output(Relay,Even_contacts)
860 !
870 PRINT "END OF PROGRAM"
880 STOP
```

After entering the above program, you can renumber the program in increments of 10 using the REN command:

Type REN and then press EXECUTE

At this point, once you have modified EXAMPLE, you could go back and RUN it again. If you do not need to change any of the configuration information specified earlier, you can edit the "Configure" statement to use the "Menu_off" option.

Storing Your Program on a Disc

As you write your application program and ENTER each line, it is stored in the computer's internal RAM. This RAM is a "volatile" type of memory, that is, your program will be lost when power to the computer is turned off. In order to save any program modifications that you have made during the course of following this chapter, you must store it permanently to the disc.

To save your program make sure that you have the "Working Copy" of the 14752A CAT Programming Package disc or an initialized disc which is not write protected inserted into your drive.

Recall that after you initially loaded the EXAMPLE application program, you then loaded in all of the 14752A subprograms from the file LIBRARY. This collection of subprograms is quite large; to conserve disc space, you **MUST** delete these subprograms before storing your modified application program onto the "Working Copy" of the 14752A CAT Programming Package disc.

- To do this, type in

  DELSUB Configure TO END
  and press EXECUTE

- If you are storing your program on the 14752A CAT Programming Package disc, using the EXAMPLE application (EXAMPLE file name in the directory), type in the following:

  RE-STORE "EXAMPLE"
  and press EXECUTE

- If you are storing your program for the first time on a separate initialized disc, or the 14752A CAT Programming Package disc under a different file name, type in the following:

  STORE "FILENAME"
  and press EXECUTE

Where "FILENAME" is a name of your choice up to 10 characters in length.

Additional information about writing very large application programs, and storing them on disc is provided in Chapter 4 of this manual under the heading "Disc Configurations."

Chapter 4
Program Development

Introduction

This chapter presents additional material to help you become familiar and proficient with the Multiprogrammer hardware and CAT Programming Package software. It explains the operation of the Soft Front Panel utility program. This program provides an easy way to learn the operation and capabilities of the system. The Soft Front Panel can help you try out various sequences of I/O statements as you are developing an application program. Other subjects in this chapter, such as system statements, error handling considerations, recommendations for the program development environment, and use of the DOCUMENT utility program, and performance-related issues will be helpful to you in program development.

Using the Soft Front Panel

The 14752A Utility Programs disc includes a "Soft Front Panel" utility that allows each of the functions you have configured to be easily and interactively executed from the keyboard using menus and softkeys. When you run this subprogram, an I/O function name is selected from a menu which lists all of the names that you included in your COM/Names/ statement. When you have selected the name, the 14752A statements that can be used with the selected function appear on a second menu. Each statement for that function can be selected and executed as you choose. The Soft Front Panel is useful in learning the statements that apply to your particular functions as well as in learning how the functions operate. The Soft Front Panel can also be used to checkout and troubleshoot the cabling for function-to-function communications as well as electrical connections to your the application. The I/O function edge connector signals are described in Chapter 10 in the section entitled "Frontplane Signals and Timing".

Calling the Soft Front Panel

There are two ways of executing the Soft Front Panel. If you have come here from Chapter 3, it was by pressing the "Front_p" softkey which appears in the Exit Menu screen (see Figure 3-11) of the Configuration Phase.

The other way you can activate this utility is by typing from the live keyboard mode of BASIC:

CALL Front_panel
followed by EXECUTE

In order to do this, however, your I/O function names must already be configured. This means that you must have previously executed an application program which includes at least your "COM/Names/" statement and "Configure" statement. If you try to call the Soft Front Panel before the I/O Function names are defined, you will get:

ERROR 145-May not build COM at this time

You will also get this error if you call the front panel after you have executed a SCRATCH C or SCRATCH A command, because they clear out the function names stored in COMmon.

The Named Function Selection Menu

As the Soft Front Panel software is being loaded, status messages will be displayed on the screen. When this process is completed, the Named Function Selection menu will appear.

Each name from the COM statement(s) which you used in your application program will be listed in this menu. If you have used many names in your application program, then the menu will be several pages long. If there are more than one page of names, PrevPage and NextPage softkeys will appear in the softkey area on each page to allow you to go from page to page. Use the computer knob or arrow keys to position the cursor on a name and press the Select softkey to select a name. Using the soft front panel menus is very similar to using the Configuration menus described in Chapter 3. The following softkeys appear on the Named Function Selection Menu (see Figure 4-1):

Select — Pressing this key will select the Name at the cursor's present location. Only I/O function names which have been configured can be selected. See description below.

SysInit — Pressing this key executes a System_init statement causing all named functions to return to the conditions that were selected during the configuration phase.

MENUER — Use this key to exit the soft front panel and return to the Welcome page (Figure 3-5) of the configuration menus. This key is useful if you have forgotten to configure a name or reconfigure a name.

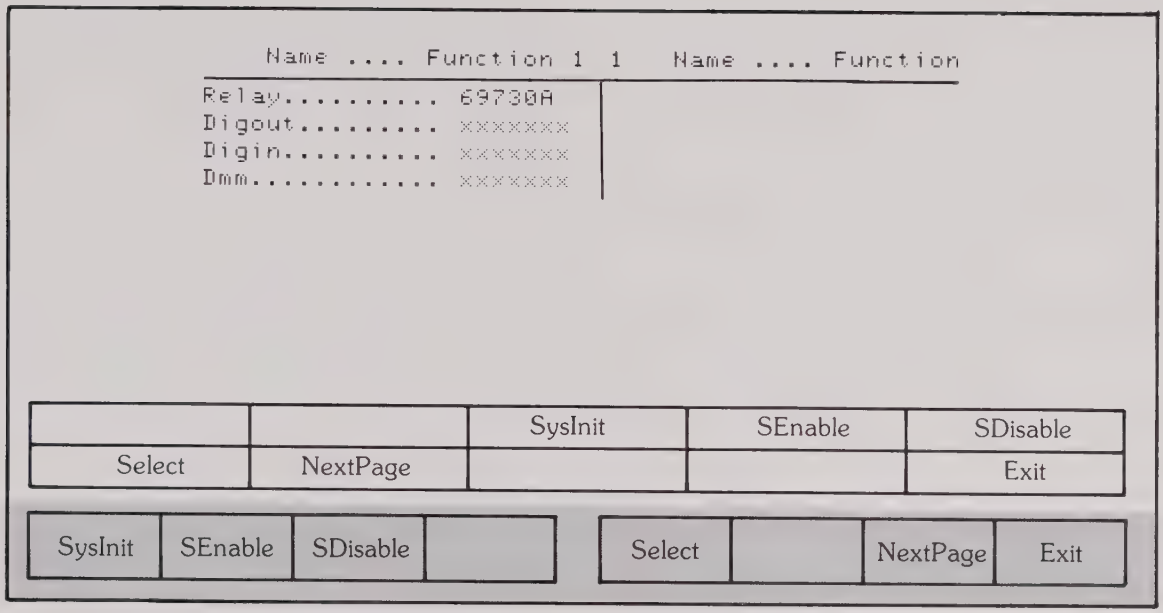


Figure 4-1. Soft Front Panel, Named Function Selection Menu

SEnable Pressing this key executes a System_enable statement. As explained later in this chapter, it enables the output of all output functions in all Multiprogrammers that contain at least one function that was configured.

SDisable Pressing this key executes a System_disable statement. As explained later in this chapter, it disables the outputs of all output functions in all Multiprogrammers that contain at least one function that was configured.

EXIT Use this softkey when you have finished using the soft front panel. If the Front Panel was called from the Front_p softkey at the Exit Menu screen of the Configuration Phase, you will resume executing your application at the line following the "Configure" statement. If the Front Panel was executed via a keyboard call, you will return to the live keyboard mode.

All names included in the COM statement(s) will be listed in the menu but only those that have been configured will have an entry adjacent to them in the Function column. The menu shown in Figure 4-1 lists all of the names that are included in the COM statements in the EXAMPLE program of Chapter 3. Note that on this menu, only the name "Relay" has been configured. Position the cursor next to the name "Relay" and press the Select softkey; the Statement Selection Menu (see Figure 4-2) for the 69730A Relay Actuator function which is named "Relay" will appear.

If you position the cursor on any of the other names (see Figure 4-1) which have not been configured ("XXXXXX" appears next to the name in the Function column), then the computer will beep and the Statement Selection menu will not appear. Consequently, if no names are configured, you cannot use the Soft Front Panel utility at this time. If this is the case, press the MENUER softkey to go back to the configuration menus.

The Statement Selection Menu

The Statement Selection Menu gives a listing of all the statements that can be used to program a particular named I/O function (or named Buffer, Timer or Scanner multiple card function.) As described in the previous paragraph, this menu will only appear if the selected name has been configured. For example, Figure 4-2 shows the Statement Selection Menu for the 69730A Relay Actuator function which has the name "Relay". This menu lists the 14752A statements (Init, Output, Preset, Start, etc.) that can be used to program the 69730A function. Use the computer knob or arrow keys to position the cursor on a statement and press the Select softkey to make the selection.

The functions provided by the Select, Enter, and NextName softkeys on the Statement Selection Menu are described below. The remaining softkeys on this menu provide the same functions as described previously for the Named Function Selection Menu.

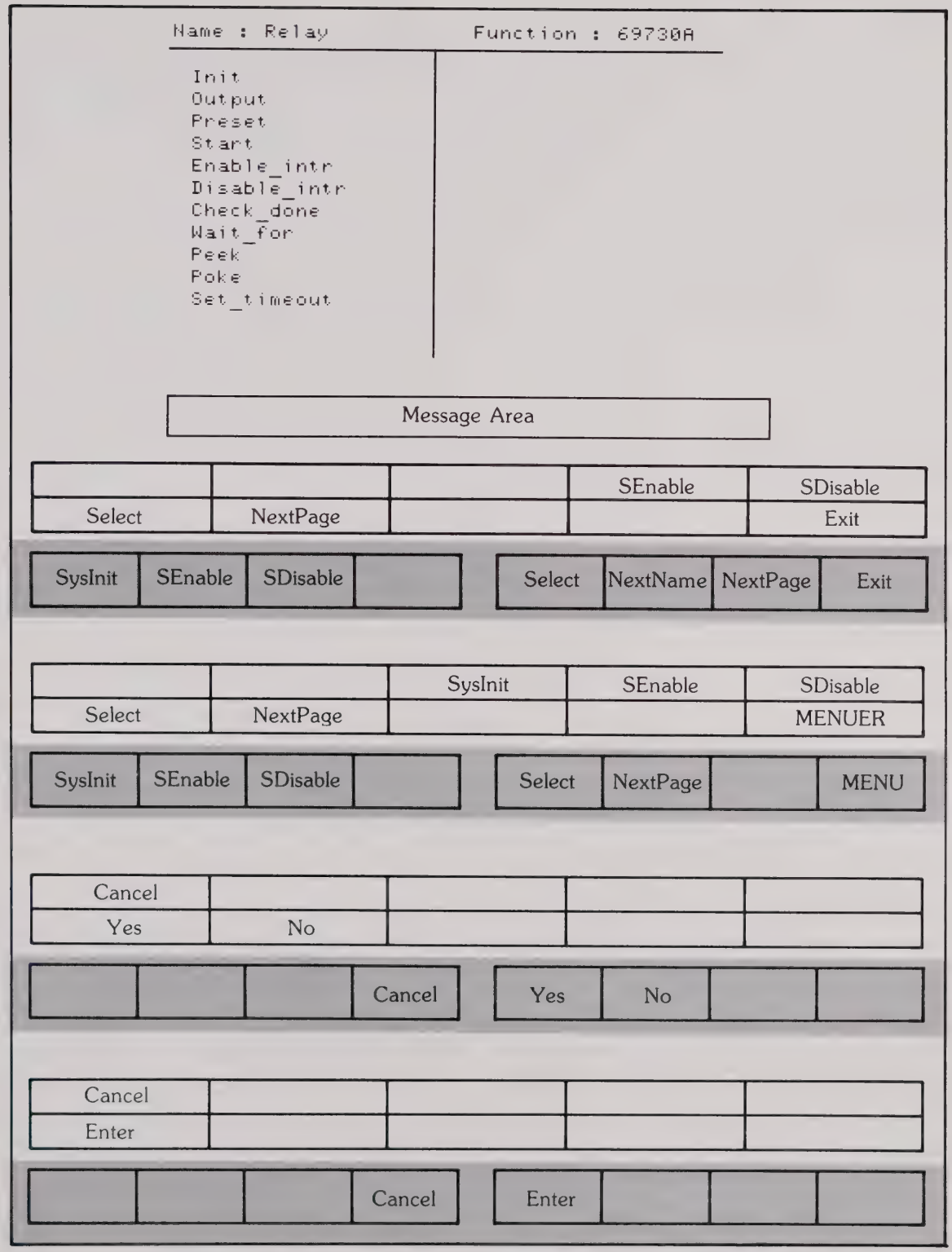


Figure 4-2. Soft Front Panel, Statement Selection Menu

Select

For statements such as Init(Name) and Start(Name) which only have the functions name in their syntax, you immediately execute these statements when you press the Select softkey. For example, if you select Init, the Relay function is initialized according to the description of the "Init" statement in Chapter 8 for the Relay Actuator function. To indicate that the function has been initialized, the following message appears on the Statement Selection Menu:

"Relay" has been initialized

For other statements, such as Output(Name,Value) and Preset(Name,Value), which have a "Value" parameter, you will see a message on the screen when you press the Select softkey. For example, if you select the Output statement, the following will appear:

Enter value to be output

Enter

Assuming that you type in the value "100", pressing the "Enter" softkey will output this value to the Relay function. Remember that the value you enter will be scaled according to the data conversion option that you specified on page 2 of the 69730A configuration menu (see Figure 3-10). The value is also subject to run Time Limit Checking if that option has been activated on the configuration menu page. In this example, when you press the Enter softkey, the value 100 is output to the edge connector of the Relay function and the following message appears on the Statement Selection menu.

100 has been output to "Relay"

Yes, No

Statements that require a yes or no response to an inquiry.

Cancel

Cancel aborts the statement, no action is taken.

NextName

Use this softkey if you want to select another named function from the Soft Front Panel. Pressing this key returns you to the Named Function Selection Menu.

There are 14752A statements which allow optional parameters to be addressed, however, the Soft Front Panel does not support optional parameters. For instance, you can only select one name from the Soft Front Panel, however, optional parameters can be specified by a live keyboard CALL to the particular statement.

Data values read from an I/O function are displayed on a single line at the bottom of the screen. Normally this is not a problem; but it can be when you are doing an "Input_iblock" or "Input_rblock" statement from the Soft Front Panel and you have a large number of data values to be read. In this case, you can stop the scrolling of the display by pressing any key on the keyboard; pressing it again will cause the data presentation to resume.

Each of the statements displayed on the Soft Front Panel (except Peek and Poke) are described in the applicable function chapter; (e.g., the 69730A Relay Actuator Function is described in Chapter 8). Information concerning the Peek and Poke statements is given in Chapter 10 of this manual.

When you have finished executing statements for a particular I/O function, you can either press the NextName softkey to select another named function or exit the Soft Front Panel as will be described.

Front Panel Errors

14752A errors can occur during Soft Front Panel operations just as they can occur during execution of your program. If the operation you have selected results in an error, the computer will beep and display a one-line message identifying the problem. The messages which are displayed are discussed later in this chapter under "14752A Error Messages".

All errors which occur during operation of the Soft Front Panel are handled locally; that is, any "ON ERROR" statement that you may have executed in your program gets overridden by the error handler of the front panel utility. Your error handler is reactivated, however, when the front panel is exited.

Interrupts and the Soft Front Panel

There are several points worth noting with regard to how I/O function interrupts interact with the Soft Front Panel.

First of all, interrupt servicing linkage (the "ON INTR" statement) and service routine software (the "User_handler" subprogram) which you have set up before calling the Soft Front Panel remain active while the Soft Front Panel is in use. Interrupt requests from any of the devices controlled by the 14752A software will be processed as you have specified in your "User_handler" subprogram.

In addition, the Soft Front Panel cannot be used to execute an "ON INTR", "OFF INTR", "Enable_sc_intr", "DISABLE_SC_INTR", "ENABLE INTR", or "DISABLE INTR" statement; they can only be executed by your application program.

You should be careful, if your "User_handler" contains "PRINT" statements which write to the display; the Soft Front Panel software cannot anticipate this, and you can find the Soft Front Panel's messages being overwritten and/or scrolled up when an interrupt occurs. It is best to use the "DISP" statement.

The Soft Front Panel does, however, allow you to execute the 14752A "Enable_intr" and "Disable_intr" statements for all functions to which they apply, so you can use this utility program to help debug your interrupt service software for specific I/O functions.

Exiting the Soft Front Panel

You have two softkey options you can choose when you are through using the Soft Front Panel utility:

MENUER

This key returns you to the "Welcome Menu" page (Figure 3-5) of the Configuration Menus. The configuration process is described in detail in Chapter 3. If the front panel was called form the "Front_p" softkey of the Exit Menu softkeys, you will return to the Configuration Menu with your current configuration information.

This key is useful if you discover you forgot to configure, or misconfigured, any of the I/O function names used in your application program.

EXIT

This key terminates the operation of the Soft Front Panel and continues program execution.

If the front panel was executed by a live keyboard CALL command, you will return to live keyboard mode.

If the front panel was called with the Front_panel statement, the execution of your application program would resume at the next line following the statement.

The "DOCUMENT" Utility

The 14752A Utility Programs disc includes a "DOCUMENT" utilty that allows you to list the complete configuration file for each name in the COM statement that has been configured. This utility provides you with a quick and efficient way of obtaining menu configurations for each name in a given configuration file. This is useful if you continually change your configurations and need to keep records for later reference. This utility can be used with the 14750A, 14751A, and 14752A to show the contents of a configuration file.

Loading "DOCUMENT"

The configuration file dump program can be accessed by loading and running the BASIC program named "DOCUMENT" which is stored on the Utility Programs disc. Note that the system software must be loaded into the computer prior to loading and running "DOCUMENT".

To load "DOCUMENT", place the 14752A "Utilities Disc" into the default disc drive of the computer you are using , then type:

Load "DOCUMENT"
and press EXECUTE

Wait for the run light to go out and press RUN. Next, remove the Utility Programs disc from the disc drive and insert the 14752A CAT Programming Package disc (Library disc), or the disc with menu__data stored on it, into the drive an close the door.

NOTE

14750A or 14751A Configuration files require their corresponding menu__data file. Now when you are ready:

Press CONTINUE

Using the Softkeys

You are now ready to use DOCUMENT. First, press the File name softkey to enter the configuration file specifier. Next, use the remaining softkeys that appear on the screen to obtain a listing of your configuration file (see Figure 4-2a). The following is a list of these softkeys:

File name — Pressing this key will allow you to enter the configuration file specifier.

CRT* — Pressing this key will select the listing device to be the CRT. The actual list will occur after the List softkey is pressed. Note that you may alternately pause/re-start the list with any key.

PRINTER — Pressing this key will select the listing device to be the printer at select code 701. The actual list will occur after the List softkey is pressed.

Note: an asterisk indicates which of the two devices (CRT/PRINTER) is the current dump device. The default is the CRT.

LIST — Pressing this key will cause either the CRT or PRINTER to list the configuration for each name. The default List device is the CRT.

STOP — Pressing this key will abort any action occurring at that time.

EXIT — Pressing this key will cause the Document utility program to terminate.

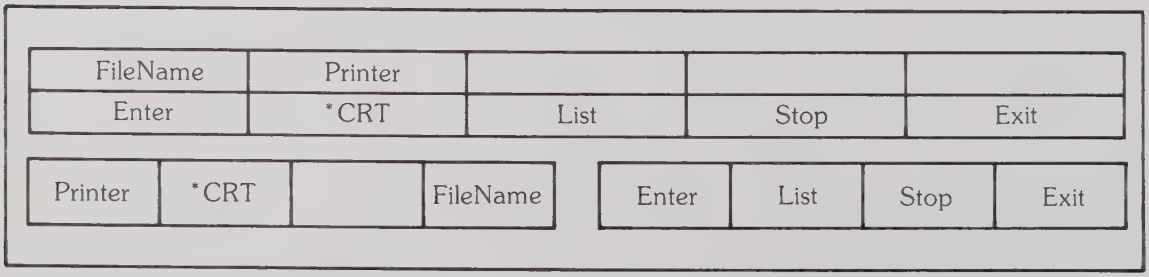


Figure 4-2a. Soft Front Panel, Document Utility Menu

Keyboard Control of 14752A Statements

In the course of your program development cycle with the 14752A, there will be times when you wish to have direct keyboard control of the I/O operations associated with your function names. There are two particular topics that merit discussion in this area.

Keyboard CALL's of 14752A Statements

In the previous section, it was discussed how the Soft Front Panel utility program can be CALLed directly from the keyboard of the computer. Since each statement of the 14752A CAT Programming Package is, in reality, a subprogram, each can be CALLed in the same manner.

To execute a 14752A statement, type "CALL" followed by the name of the statement and its parameters then press EXECUTE. For example, you could output a new value to the function name "Relay" by typing:

CALL Output (Relay, 64)
then pressing EXECUTE

To input a value from the name "Digin", type:

CALL Input (Digin,A)
then press EXECUTE

The value read from "Digin" would be stored in the variable "A". Note that "A" must have been declared explicitly in your program in order for it to be used in a keyboard CALL. If you plan to use this keyboard CALL feature during the course of your program development, you should include a variable declaration statement in your program, to give yourself several variables you can use for this purpose:

REAL A,B,C,D

Note that many of the same restrictions apply to CALLing an I/O statement as apply to CALLing the Soft Front Panel. The I/O function names must already have been defined by a program which includes, at minimum, the "COM/Names/" statement and the "Configure" statement. As mentioned earlier, a SCRATCH C or SCRATCH A command will clear out COMmon.

If you try to CALL a 14752A statement when the I/O function names either do not exist in "COM/Names/" or are undefined you will get either:

ERROR 812-"Undefined Name Reference"
or
ERROR 910-"Ident not found in context"

Stopping Execution of a 14752A Statement

Most 14752A statements are implemented as compiled subprograms, or "CSUB's". By editing your program after you have loaded all the subprograms from the LIBRARY file, you will notice a BASIC subprogram header line with the keyword CSUB for every 14752A statement. The code itself, since it is written in PASCAL and/or Assembly Language, it is not visible to you.

Stopping execution of a CSUB is different than stopping execution of other BASIC statements or subprograms. The only way of stopping a CSUB is to do a BASIC "Reset" (Shift-PAUSE) or (Shift Reset). Neither "PAUSE", "CLR I/O", "STOP", nor "BREAK" will work in this situation. Note that there are a few 14752A statements, "Wait_for" being the most notable, which are implemented partly as BASIC subprograms and thus will behave accordingly.

The "Timeout" feature of the 14752A language was included to provide a smooth exit mechanism for those statements which are most subject to incomplete I/O transactions.

NOTE (Applies to 6944A only)

Resetting the computer causes a system disable. You need to Re-enable the system (System_enable) if you do not re-run your program from the beginning.

14752A System Statements

The class of 14752A statements which perform operations on an entire group of I/O function Names at a single time are referred to as System Statements. The following three statements belong to this group.

System__disable

This statement is used to simultaneously "disable" the output of all output functions in all the 6944A and 6942A frames in your CAT system. The definition of "disabled" is function dependent. For example, all relays on the 69730A Relay Actuator function are opened, while the output voltage of the 69720A D/A Converter is set to 0 volts. The System__disable statement deactivates the hardware line "SYE" System Enable on the Multiprogrammer backplane. Information on the effect of "System__disable" for specific I/O card scan be found in the individual I/O card manuals.

System__enable

This statement is used to simultaneously "enable" the outputs of all output functions in all the 6944A and 6942A frames in your CAT system. By enabling these functions, you are allowing the values you programmed with the Init, Output, and Preset/Start statements to be present at the edge connector of each I/O card. The System__enable statement activates the SYE hardware line in the Multiprogrammer. The 14752A system "wakes up" with SYE enabled; this statement does not have to be executed unless you have executed a "System__disable" command somewhere in your program as in the Soft Front Panel.

System__init

This statement is used to initialize all the named I/O functions. They are initialized in the order that the names appear in the COMmon statement. At initialization the function's interrupt is disabled and the operating mode (if applicable) and data conversion options specified in the menus are set. Output functions can be programmed to some initial value and Run Time Limit Checking can be selected. This System__init statement **must** be located ahead of the Enable__intr statement in your program.

14752A Errors

There are two distinct types of errors that can occur with the 14752A CAT Programming Package Software: Configuration-Time Errors and Run-Time Errors.

Configuration-Time Errors

The first type of errors are the ones you get while you are interacting with the Configuration Menus; they occur when you enter an incorrect value or parameter in a menu. In most cases, the error message description appears in the status box at the bottom of the menu screen as described in Chapter 3. The menus are usually friendly enough to point you to where the error occurred, and interactive enough to allow you to fix the error with a minimum of delay.

Run-Time Errors

The second type of errors are known as "Run-Time Errors". They can occur during the execution of any BASIC or 14752A statement. Unless you have made special provisions in your program for handling these run-time errors as they occur, the computer will beep, an error message will be displayed on the bottom line of the screen, and execution of your program will be "PAUSE'd." Pressing the CONTINUE key will cause the program to go back and try to re-execute the statement in which the error occurred. In most cases, you need to edit the program and then run it again.

14752A generates error codes in the 800's. Error codes in the high 300's may or may not be caused by the 14752A software. All other error codes are generated by the BASIC Language; a list and description of each can be found in the Appendix C of the "BASIC Programming Techniques".

In addition to the error code on the bottom line of the screen, 14752A run-time errors also generate a three-line message on the display. The message is usually of the form:

```
14752A: (Error__descriptor)
        detected by (Statement__name) for model 697XX
        (Function__model)
```

where:

"Error__descriptor" is a description of the error that occurred ("Timeout Error", "Wrong Memory Mode", etc.)

"Statement__type" is the 14752A statement in which the error occurred (for example, "Output", "Input", "Wait__for", etc.)

"Function__model" is the I/O function hardware for which the error occurred (for example, "Relay Output", "Digital Multimeter", etc.)

All 14752A error messages are described in Table 4-1. Appendix B provides a convenient listing of all 14752A error codes.

Table 4-1. 14752A Error Messages

Error Code	Descriptor and Explanation
801	Unsupported Model Error The I/O statement is not applicable to the function represented by "Name". Occurs only at run time.
802	Soft Limit Exceeded Limit Checking was ON, and the data value to be output was outside the range defined by the Low and High Limit. Applies only to Output statements. Occurs only at run-time.
803	Timeout Error The function did not complete its operation within the time allotted for "Timeout". Function complete is tied to the state of the EOP (End of Process) signal line.
804	Array Size Too Small The statement requested more data transfers than the number of elements allocated for the array. This applies to all functions that use the Input_xblock statements. Occurs only at run-time.
805	Reserved-5 This is a system error that should not occur during normal operation. If this message appears, contact your HP Systems Engineer.
806	Illegal Data Conversion Option This is a system error that should not occur during normal operation. If this message appears, contact your HP System Engineer.
807	Illegal Limit Checking Option This is a system error that should not occur during normal operation. If this message appears, contact your HP System Engineer.
808	Interrupt Device Not Found An interrupt occurred but no requesting device was found. Occurs only at run-time.

Table 4-1. 14752A Error Messages (continued)

Error Code	Descriptor and Explanation
809	Reserved-9 This is a system error that should not occur during normal operation. If this message appears, contact your HP Systems Engineer.
810	Illegal Vector Number The number specified for "Vector" in an "Enable_intr" statement was outside its legal range of 1 through 32767. Occurs only at run-time.
811	Names Not All At Same Address The I/O functions represented by the various Names in a "Start" statement were not all located at the same Multiprogrammer Interface Select Code. Occurs only at run-time.
812	Undefined Name Reference The "Name" specified in the statement was not configured, or is not contained in the "COM/Names/" statement. Occurs only at run-time. (NOTE: Error 910 may also occur for the same reason.)
813	Names COM Block Not Found There is no "COM/Names/" statement in the application program. Occurs only at configuration-time.
814	Illegal Mode ID This is a system error that should not occur during normal operation. If this message appears, contact your HP Systems Engineer.
815	Use of unINITialized Function An attempt was made to perform an I/O operation with "Name" before an "Init(Name)" or "System_init" statement was executed. Occurs only at run-time.
816	Illegal Device Type This is a system error that should not occur during normal operation. If this message appears, contact your HP System Engineer.

Table 4-1. 14752A Error Messages (continued)

Error Code	Descriptor and Explanation
817	14752A System Error This is a system error that should not occur during normal operation. If this message appears, contact your HP System Engineer.
818	Illegal Data Value The data value specified is outside the range supported by the function. Applies only to Output statements. Occurs only at run time.
819	Illegal Initial Function This is a system error that should not occur during normal operation. If this message appears, contact your HP System Engineer.
820	Illegal Initial Range This is a system error that should not occur during normal operation. If this message appears, contact your HP Systems Engineer.
821	Illegal Edge In Define This is a system error that should not occur during normal operation. If this message appears, contact your HP Systems Engineer.
822	HP-IB Bus Timeout An HP-IB bus transaction did not complete. Occurs when the 3478A is not present in the system, or is set to an address different from that specified in the menus for the "Name". More information can be found in Chapter 3 in the section entitled "Leaving the Exit Menu". Can also occur when a "Get" statement is issued to the 3478A when data is not available. 6942A Multiprogrammer not on line.
823	Illegal Initial Mode Specified This is a system error that should not occur during normal operation. If this message appears, contact your HP Systems Engineer.
824	Illegal Data Type Specified This is a system error that should not occur during normal operation. If this message appears, contact your HP Systems Engineer.

Table 4-1. 14752A Error Messages (continued)

Error Code	Descriptor and Explanation
825	Wrong Memory Mode The current mode of the 69790B/69791A Memory function does not support the operation requested by the statement. When doing an "Input_xblock" operation, the function must be in either the "FIFO In" or "Recirculate IN" mode; when doing an "Output_xblock" operation, the function must be in either the "FIFO Out" or "Recirculate Out" mode. Applies only to the 69790B/69791A Memory function. Occurs only at run-time.
826	Illegal Error Number This is a system error that should not occur during normal operation. If this message appears, contact your HP Systems Engineer.
827	Low Limit > High Limit The low limit specified in the menus is higher than the high limit. Occurs only at configuration-time.
828	Start + Length Too Long The sum of the values of "Start" plus "Length" in the menus was greater than 16, the maximum number of bits per card. Applies only to I/O functions with variable-length names: the bit-oriented cards. Occurs only at configuration-time.
829	Reserved-29 This is a system error that should not occur during normal operation. If this message appears, contact your HP Systems Engineer.
830	Multiprogrammer Not On-Line Occurs when the specified Multiprogrammer is not present in the system, or is set to select code different from that specified in the menus for "Name". More information can be found in Chapter 3 in the section entitled "Leaving the Exit Menu".

Table 4-1. 14752A Error Messages (continued)

Error Code	Descriptor and Explanation
831	Illegal Initial Trigger This is a system error that should not occur during normal operation. If this message appears, contact your HP Systems Engineer.
832	Low Limit Too High Limit Checking is ON, and the Low Limit specified is higher than the maximum data value supported by the funciton. Occurs only at configuration-time.
833	High Limit Too Low Limit Checking is ON, and the High Limit specified is lower than the minimum data value supported by the function. Occurs only at configuration-time.
834	Illegal Frame Error The frame number specified must be in the range of 0 through 7. Occurs only at configuration-time. (NOTE: This error may also appear as "Out of bounds entry. Limits are 0 to 7").
835	Illegal Slot Error The slot number specified must be in the range of 0 through 15 (0 through 14 for the 69790B/69791A Memory function with no 69792A expansion cards). Occurs only at configuration time. (NOTE: This error may also appear as "Out of bound entry. Limits are 0 to 15").
836	Interface Card Not Present The 98633A Multiprogrammer Interface card or 98624A HP-IB Interface card is not present at the specified select code of the computer. More information can be found in Chapter 3 in the section entitled "Leaving the Exit Menu". (NOTE: This error message may also appear when you get "Error 388".)

Table 4-1. 14752A Error Messages (continued)

Error Code	Descriptor and Explanation
837	Specified Card Not In Slot An incorrect I/O card, or no card, was found in the slot associated with "Name", instead of the model which it expected to find. More information can be found in Chapter 3 in the section entitled "Leaving the Exit Menu".
838	Illegal Name: Not a Scalar REAL The "Name" specified was not of type REAL. "Name" must be defined in the "COM /Names/" statement, and may not be of type INTEGER.
839	Illegal Initial Auto Zero This is a system error that should not occur during normal operation. If this message appears, contact your HP System Engineer.
840	Illegal Initial Interrupt Mask This is a system error that should not occur during normal operation. If this message appears, contact your HP Systems Engineer.
841	Missing Address Buffer Memory The 69790B/69791A Memory function specified as the Address Buffer memory for a Scanning function configured for "random scan" mode is not present. Occurs only at configuration-time.
842	Illegal Initial Start Bit This is a system error that should not occur during normal operation. If this message appears, contact your HP Systems Engineer.
843	Illegal Bit Number The bit number specified was less than 0 or greater than 15 minus "Start Bit". Applies only to the bit-oriented statements of the 69776A Interrupt Card. Occurs only at run-time.
844	Illegal Number of Digits This is a system error that should not occur during normal operation. If this message appears, contact your HP Systems Engineer.

Table 4-1. 14752A Error Messages (continued)

Error Code	Descriptor and Explanation
845	Wrong Scan Mode The wrong "scan" statement was executed. The statement used must be the one that matches the scan mode specified in the menu at configuration time, either "random" or "sequential". Applies only to Scanning functions This error occurs only at run-time.
846	No Range Info For Data Conversion You **MUST** supply range values for the Translate and Demultiplex statements with the model 69759A A/D card. You must also have range readback set when using a Get statement with the range of the 69759A A/D card set to external. Occurs only at run-time.
847	Illegal Range Value An illegal range value has been specified. For the 3478A, this is a value other than .030, .300, 3, 30, 300, 3k, 300k or 30M. For the Model 69759A A/D card, this is a value other than 10,1, .1,100 or 5, .5, .05, 50. Occurs only at run-time.
848	Illegal Select Code The specified select code is outside of the allowable range, 8 to 31. Occurs only at run-time.
849	Interface Must Be A 98633A The Specified select code must refer to a 98633A Multiprogrammer Interface card. Occurs only at run-time.
850	Configuration Error This is a system error that should not occur during normal operation. If this message appears, contact your HP Systems Engineer.
851	Buffer Must Have Front End The buffer menu **MUST** reference a front-end card. Occurs only at configuration-time.

Table 4-1. 14752A Error Messages (continued)

852	Buffer Has An Incorrect Front End The buffer menu can only reference a model 69720A, 69721A, 69751A, or 69759A. The selected name **MUST** refer to one of these models. For the front-end 69759A card in a multiplexed configuration, all names selected must refer to this model type and no two names in the list can refer to the same physical card. Note that all names must be selected in numeric order; no selections can be skipped. Occurs only at configuration time.
853	A/D Has An Incorrect Scanner The 69751A/69759A A/D card can only reference a named Scanner or Scan Controller. Occurs only at configuration time.
854	No Digout Controller in Timer In order to execute these statements, you must specify a Digital Output Controller card in the Timer menu. Occurs only at run-time.
855	Full Scale Range Mismatch Two names refer to the same A/D card and each has a different specified full-scale range. Occurs only at configuration-time.
856	Transmission System Failure Be sure the Multiprogrammer is turned on and the extenders are turned on before the Multiprogrammer. Check for defective or loose interfacing cables and be sure the interface or transmission cards are securely plugged in.
857	I/O Card Test Failure This indicates that one of the cards you have configured has failed the I/O card test. Occurs at configuration-time.
858	Array Size Mismatch The Integer array and Real array used in the Translate statement are not the same size. Occurs at run-time.
859	Too Many Parameters The optional Range array parameter can only be used with the Translate statement for a model 69759A card. Occurs at run-time.

Handling Run-Time Errors

During program development, and possibly even in some applications, it does not harm or is even advantageous for a program to stop when an error occurs. The error often is caused by a flaw in the logic or flow of your program, and you wish to be informed of it so that it can be corrected.

There are other applications, however, where you wish to take some well-defined actions in case a run-time error occurs. An example might be the one where you have just applied the driving voltage to the buzz saw in a paper mill and an error occurs in the very next statement of the program which controls the machinery. The program stops; but since the voltage is still applied, the saw keeps going and it winds up slicing everything in its path!

This type of situation is the one which provides the best application for run-time error handling.

Writing an Error Handler

The mechanics of programmatically handling run-time errors is easy. Series 200 and 300 BASIC has an "ON ERROR" statement that allows you to specify what statements should be executed if a run-time error occurs in your application program.

There are two versions of the "ON ERROR" statement that are useful with the 14752A software package: "ON ERROR RECOVER" and "ON ERROR CALL".

NOTE

*It is recommended that you do **NOT** use the other two variations: "ON ERROR GOTO" and "ON ERROR GOSUB". The problem with them is that they are context-sensitive. In Series 200 and 300 BASIC, each subprogram is a "context", independent of other subprograms and the main program. The error-handling linkage set up by the GOTO and GOSUB versions of the "ON ERROR" statement remain active only within that context; that is, any errors which occur outside of the context in which the "ON ERROR" statement was executed will not be handled by it.*

On the other hand, linkages set up by the RECOVER and CALL versions are global; they remain active even when the program context changes.

Shutting Down the I/O System When an Error Occurs

This is the solution for the buzz-saw problem!

To begin with, you should configure each of your output functions so that they each get set to a "safe" state when they are INITialized. This will allow you to use a "System_int" statement to get back to the safe state within your error handler. (Remember that you will have to re-enable interrupts after each System_init statement).

Note that this concept is similar to that of the "System_disable" statement, but with one major difference: you define the "safe" state for your application yourself, rather than being forced into a fixed set of conditions defined by the I/O hardware.

There are two sections to the error handler code: the Error Handler Setup and the Run-Time Error Handler. The EXAMPLE application program (Figure 3-1) in Chapter 3 includes Error Handler and Run-Time Error Handler sections. These sections are repeated below for this discussion.

```
670 ! ------ERROR HANDLER SETUP------
:
:
780 ON ERROR RECOVER Error_handler
790 Set_error_print (0)
:
810 ! ------BODY OF APPLICATION PROGRAM------
:
900 ! ------RUN-TIME ERROR HANDLER------
:
:
960 Error_handler:!
:
:
1010 PRINT "ERROR OCCURRED"
1020 PRINT ERRM$
1030 STOP
:
1070 END
```

The "ON ERROR RECOVER Label" version of the statement (line 780) is used to set up the linkage. When a run-time error occurs anywhere within the program, the statements following the label "Error_handler" will be executed.

The "Error_handler" statements are located in the main program context, before the "END" statement (lines 900 through 1030). In this example, the "System_init" statement used to return the output functions to their safe state would get inserted at line 1000.

Correcting Errors When They Occur

The other use of run-time error handling requires quite a bit more effort and forsight to apply. The concept is that when an error occurs you can figure out what you have to do to correct it "on-the-fly", then go back and re-execute it correctly.

In this type of application you would use the "ON ERROR CALL Subprogram" statement in the Error Handler Setup block.

When an error occurs, program flow would pass to the named subprogram. Based on information about the error you can define what action to take for a given program line. Your subprogram **must** be located **after** the main program's END statement but before the 14752A Library of CSUB's. When you SUBEXIT or SUBEND the subprogram, you will go back to the statement which originally caused the error and re-execute it.

Since this is a more complicated and more application-dependent use of run-time error handling than the previous case, it was not built into the EXAMPLE program, and is left for you to develop.

The "Set_error_print" Statement

There is one additional 14752A statement that can be useful during run-time error handling. This statement enables or suppresses the generation of the 14752A error message whenever an 800-series error occurs. If you are handling errors yourself, you may not want the error message to appear on the display.

The statement (line 790 of EXAMPLE) has the form:

Set_error_print(Flag)

A flag of 0 is used to suppress the error message; a flag of 1 enables the generation of error messages.

NOTE

A "Set_error_print(1)" is executed automatically when you exit either the configuration phase or the Soft Front Panel.

Software Performance

Part V of the Specifications in Chapter 1 give values for execution times of the most commonly used statements of the 14752A software. If you are concerned with I/O throughput, the following discussion will be of interest.

I/O Throughput

The execution time of any I/O operation can be broken into two independent parts:

1. The time it takes for the software to decode your statement, massage the data, initiate the I/O operation, detect its completion, and return for the next statement.

2. The time it takes for the I/O hardware, and the application to which it is connected, to complete the requested operation and so inform the computer.

The execution times specified in the table are typical values for part 1, and assume that the time for part 2 is zero.

In some cases, the time it takes for the hardware is very small compared to the time it take the software. This is the case, for instance, in an Output statement to a 69731A Digital Output Card on which the "gate" and "flag" signals (see Chapter 10) are wired together. This also applies to most of the statements which do not cause the card-edge outputs to change, such as Enable_intr, Disable_intr, Preset, Get, etc.

In other cases, the time for the hardware is a constant which is large compared to the time for executing the software. This is the case for the 69730A Relay Output Card, where the time for part 2 is the 6 milliseconds it takes for the relays to settle in an Output statement.

In still other cases, such as ones where the "gate" signal is wired to external hardware which in-turn generates the "flag" signal, the time for the hardware is application-dependent, and should be measured in your system if it is critical.

Block Data Transfer Times

Part V of the Specifications in Chapter 1 also lists typical times for the various operations performed during the transfer of a block of data in a single statement. Statements which perform such transfers include the Input_rblock, Input_iblock, Output_rblock, and Output_iblock statements.

The sequence of operations is as follows for block output and block input statements:

Block Output	Block Input
Block transfer entry overhead *Data conversion time (per word) Output Data transfer time (per word) Block transfer exit overhead	Block transfer entry overhead Input Data transfer time (per word) *Data conversion time (per word) Block transfer exit overhead

* NOTE: data conversion times apply only to "rblock" transfers.

For block input operations, all the words are input before any of them are converted (data conversion is done for "rblock" transfers only). That is, if you have requested 4096 words to be read, the 4096 words will be input and stored, after which they will all be converted to the proper data conversion units ("standard", "user", or "none").

For REAL block output operations, data transfers are done in blocks of 1024 maximum. If you have requested 4096 words to be output, 1024 data conversion operations will be performed (data conversion is done for "rblock" transfers only), then these 1024 values are output. This procedure will be repeated four times (4 x 1024 = 4096). The algorithm was adopted to conserve Heap space.

Interrupt Service Times

Interrupt service performance is largely governed by the workings of Series 200 and 300 BASIC. An "ON INTR" statement (see Chapter 10) is used to set up beforehand the linkage to the 14752A "Intr_handler" subprogram, which in turn will call your own service routine "User_handler" when an interrupt occurs.

After receipt of an interrupt, BASIC can pass control to your service routine only when the statement it is currently executing gets completed. For instance, if you had just begun executing a "WAIT 10" statement when an interrupt occurred, you would have to wait the 10 seconds for the statement to complete before servicing of the interrupt began. Because of this, the Interrupt to "User_handler" Entry time in Part V of the Specifications in Chapter 1 must be considered a **MINIMUM** time.

This time is the delay from when the interrupt occurs until you begin executing the first statement in your "User_handler" subprogram.

Once one interrupt occurs, further ones cannot occur until you re-execute the "Enable_sc_intr (Select_code)" or "ENABLE INTR Select_code; 2" statements(s) in "User_handler". The time necessary to service an interrupt is totally application-dependent.

The third factor in interrupt service routine performance is the time required after interrupt servicing is complete; the time from when you execute a "SUBEXIT" or "SUBEND" statement until you resume your program where it left off.

Benchmarking I/O Performance

As mentioned previously, you may desire to measure the performance of an I/O statement or group of I/O statements in your application. Included in Figure 4-3 is a sample program which can help you "benchmark" your specific application program by making use of the real-time clock in the computer. It is recommended that you execute the loops at least 10,000 times, as done in the example, to insure reasonable accuracy. It is also advantegeous, for statements to which you send a data value, to use the loop index variable to vary the data value used across its range.

```
10     COM /Names/ Name
20     Configure("Ask_me","TEST")
30     Init(Name)
40     !
50 Loop:!
60     !
70     ! Time the dummy loop,
80     ! followed by the loop with
90     ! the instruction.
100    !
110    A=TIMEDATE
120    FOR I=1 TO 10000
130    NEXT I
140    B=TIMEDATE
150    FOR I=1 TO 10000
160    ! Your I/O statement replaces this remark
170    NEXT I
180    C=TIMEDATE
190    !
200    ! Calculate the execution time
210    !
220    PRINT "TIME PER STATEMENT = ";(C-2*B+A)/10000
230    !
240    ! Measure it several times for repeatability
250    !
260    GOTO Loop
270    END
```

Figure 4-3. Benchmark Program

Expanding the Number of "Names"

The 14752A CAT Programming Package will support up to 128 I/O function names without modification. There is an absolute limit of 512 names. To expand the allowable number of names beyond 128 requires some explanation about how data storage is alloted to each name.

Each I/O function name requires between 200 and 300 bytes of storage once it is defined. The storage area for each name is allocated out of a "Heap" which is a separate COMmon area in BASIC, accessible by CSUB's.

The COM statement defining the CSUB heap is contained in the 14752A file LIBRARY. The statement reads

ISSS_heap: COM/Isss_heap/Isss_heap(4000)

Each REAL variable occupies eight bytes of memory; hence the default heap size is 32,000 bytes.

Sufficient room must be alloted by this statement to accomodate all the names you will be using in your application.

Each additional byte of memory which you reserve for use by an I/O function name is one less byte that is available for your application program. Conversely, if you have only a few names but a large program, you can reduce the size of the heap and gain more space for your program and/or variables.

To change the size of the heap, perform the following sequence of operations:

- ☐ LOAD "LIBRARY"
- ☐ Enter the EDIT mode
- ☐ FIND "Isss_heap"
- ☐ Edit the number of REAL's from 4000 to the new value, and press ENTER
- ☐ RUN (to perform the pre-run operations)
- ☐ When run light a lower right hand corner disappear:
- ☐ PURGE "LIBRARY"
- ☐ STORE "LIBRARY"

Disc Configurations

The disc files which comprise the 14752A software are shipped in a configuration which allows you to get started with the 14752A and write a small application program with a minimum of shuffling discs in and out of the drive, or copying them. However, if you are going to be developing a large application program, you will quickly find yourself running out of room on the 14752A CAT Programming Package disc. It will be necessary for you to configure your own disc for program development.

Disc Configuration for Program Development

It is recommended that you adopt the following disc file configuration for use during program development: if you are using a 9121D disc drive or a 5¼ disc drive:

These three discs should be configured by copying the appropriate PROG and BDAT files from the "Working Copy" of the 14752A onto three blank initialized discs.

No. 1 SYSTEM DISC		No. 2 DEVELOPMENT DISC		No. 3 TEST DISC	
*LIBRARY	PROG	MENUER	PROG	VERIFY	PROG
		menu_data	BDAT	CARD_TEST	PROG
		FRONT_P	PROG	SELFT	PROG
		fp_data	BDAT		
		yourconfig	BDAT		
		YOUR_PROG	PROG		
		DOCUMENT	PROG		

*Be sure to pick up the correct version of library to match the version of basic used.

The above configuration will minimize disc swapping and provide you with a large amount of room on the development disc for application program storage. Each of the files on the discs should be recorded in the order shown in the above list. Note that when you store your application program, e.g. "YOUR_PROG", to disc, you should be careful to delete the 14752A Library from the code if you have done a LOADSUB ALL FROM "LIBRARY", previously. To delete the LIBRARY, type:

DELSUB Configure TO END
and press EXECUTE.

Disc Configuration for Run-Only Applications

When you are all through developing your application program and it will be used repetitively in a production type environment, you may want to configure a "run-only" disc for this situation. At this point, your I/O function names should be completely defined, so you will no longer need to go through the configuration menus when the program is run. Likewise, you should not need the front panel on a regular basis, either.

This run-only disc will provide space for an application program approximately 1800 lines long. It should contain only two files:

1. Your configuration file
2. Your application program, including LIBRARY; if running BASIC 2.0, file "INTR2-1" should also be copied off.

You must get your configuration file onto the "run-only" disc. The easiest way to do this is to RUN your program with "Menu__on" and use the "Store" softkey of the Exit Menu to make a copy of the configuration information on a blank initialized run-only disc.

Combining your application program with LIBRARY is simple:

- ☐ LOAD Language System
- ☐ LOAD "your application program"
- ☐ LOADSUB ALL FROM "LIBRARY"
- ☐ Press RUN to pre-run your program; when it begins executing, PAUSE it.
- ☐ Insert a blank initialized disc in the drive and
- ☐ STORE "your program name"

Copying Disc Files Using the "BASIC Utilities" Software

Although you can always copy a BASIC applications program by LOADing it from one disc and STOREing it on another, and can do the same for your configuration files by means of the Exit Menu softkeys, you do not have the capability from the keyboard in all configurations to copy other BDAT files such as "menu__data" or "fp__data".

The computers, however, have utility programs that allow you to copy either an entire disc (CBACKUP) or selected files (FBACKUP).

Refer to the "BASIC Utilities Library" manual for information on how to use these facilities.

Programming Information

Chapters 5 through 9 in this manual contain functional, menu, and programming information for each of the I/O functions supported by the 14752A. Since each chapter is organized by function, you need read only those sections which describe the functions which you have purchased for your particular application.

Chapter 5
Analog Input Functions

Introduction

The Analog Input functions perform measurements of DC or AC voltage, DC or AC current, or resistance. The 14752A supports Multiprogrammer Analog Input functions as well as the 3478A Digital Multimeter.

The 69751A 33 kHz A/D Converter card provides readings at a maximum rate of 33,000 per second. The 69759A 500 kHz Converter card provides voltage readings at a maximum rate of 500,000 per second. For a 69759A, these readings can be written to a 69791A system and burst back to the Series 200 Computer using DMA (direct Memory Access).

The 69750A Scan Control card and associated Scanner cards (69752A, 69754A, and 69755A) can be used to scan up to 1024 analog input signals. The 69750A function can be used to direct one of 1024 analog signals to the input of the 69751A/69759A A/D Converter card. This is known as a Scanning A/D multiple card function and is configured in software via the Scanner linking field associated with the 69751A/69759A A/D Converter card menu (for details refer to A/D and Scan Control sections of this chapter). The A/D card (69751A/69759A) or Scanning A/D function can be combined with a 69790B Memory card or 69791A Memory Card system to form a Buffered A/D or Buffered Scanning A/D multiple card function. In addition, multiple 69759A 500 kHz A/D's can take simultaneous readings, then write these readings serially to a single 69791A buffer memory. This can be configured via the Buffer menu (refer to Chapter 9).

The 3478A is the only HP-IB device supported by the 14752A other than the 6942A Multiprogrammer. It is a 3½ to 5½ Digit Multimeter that provides high accuracy an high resolution measurement capability. It can measure DC and AC voltage, DC and AC current, as well as resistance (2 wire and 4 wire ohms). Selectable speed and resolution provide the right capability for your measurement need.

The table on the following page summarizes how the Analog Input functions, described in this chapter, can be used.

Analog Input Functions Summary

	Single Channel	69752A 64 Channel FET	69755A 16 Channel FET	69754A 32 Channel Relay	69790B 4 k Memory	69791A/792A 64 k Memory (expandable to 1 Mbyte)	Multiplex A/D	DMA (Requires 98620B DMA Controller card)
HP 69751A 33 kHz ADC	X	X ± 10.24 V max	X ± 10.24 V max	X 100 V max Dry Reed	X	X		
HP 69759A 500 kHz ADC	X	X ± 10.24 V max	X ± 10.24 V max	* X 100 V max Dry Reed	X	X	X (up to 8 69759A)	X
HP 3478 DMM	X	X ± 10.24 V max **	X ± 10.24 V max **	X 100 V max Dry Reed				

Scanning Front-end
"Scanner" Menu
Chapter 5

Buffer Memory
"Buffer" Menu
Chapter 9

* Channel to channel scan rate is limited by the 69754A.

** Update rate set by the 3478A; the voltage accuracy is limited by the 69752A/69755A.

69751A Analog-To-Digital Converter

The 69751A Analog-to-Digital Converter card measures voltage with a conversion period of 30 μs. It employs the successive approximation technique. This means that the analog voltage is sampled during a narrow window (200 ns) and then quickly converted to a 12-bit binary number. The card is preconfigured to measure in the range − 10.240 volts to + 10.235 volts. This range can be changed via switches and/or jumpers on the card. Whatever range is selected must be specified on Menu Page 1, so that the readings returned from the card have the proper values. Instructions for modifying and recalibrating the card for other bipolar and unipolar ranges are given in Appendix A of this manual and complete hardware details are given in the 69751A Operating Manual (HP Part No. 69751-90003).

Menu Entries for the 69751A A/D Converter

The configuration and initial conditions for the 69751A A/D card are specified on a two page menu. Before programming the function, you must specify various parameters in the menu. The 69751A A/D function's parameters are described as follows:

```
Name --- Adc                        Date: 21 Aug 1984
Model 69751A 33 KHz A/D             Time: 17:01:11
File Name -- Chapter_5
User ID ----
Time-stamp - First Time

                                 Page 1 of 2

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot ..........
Full-scale Range +- 10.24 Volts .... Yes


Scanner ....... None
Timeout ....... 5
```

Menu Page 1 of 2

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ...... 8
HP-IB Address .......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the 69751A A/D card. It has a legal range from 0 to 7. The default value is 0.

Slot

Identifies the location of the 69751A A/D card within the specified frame. It has a legal range from 0 to 15. The parameter has no default value; it **MUST** be filled in.

Menu Page 1 of 2 (continued)

Full-scale Range ±10.24

The full-scale input range of the 69751A is factory preset for −10.240 volts thru +10.235 volts. If this range meets your needs, leave the default answer, "Yes." If you require a different range, answer "No" to this question and refer to the 69751A Operating Manual for information on setting the range switches. Information useful in answering the next two questions, "Range" and "LSB Value," can also be found in the 69751A Operating Manual.

Range

Refers to the configuration of the input amplifer of the 69751A. A Unipolar range is one having a lower bound of zero. For example, 0 to 10 volts is a Unipolar range as opposed to −10 to +10 volts which is a Bipolar range. The default is Bipolar.

LSB Value

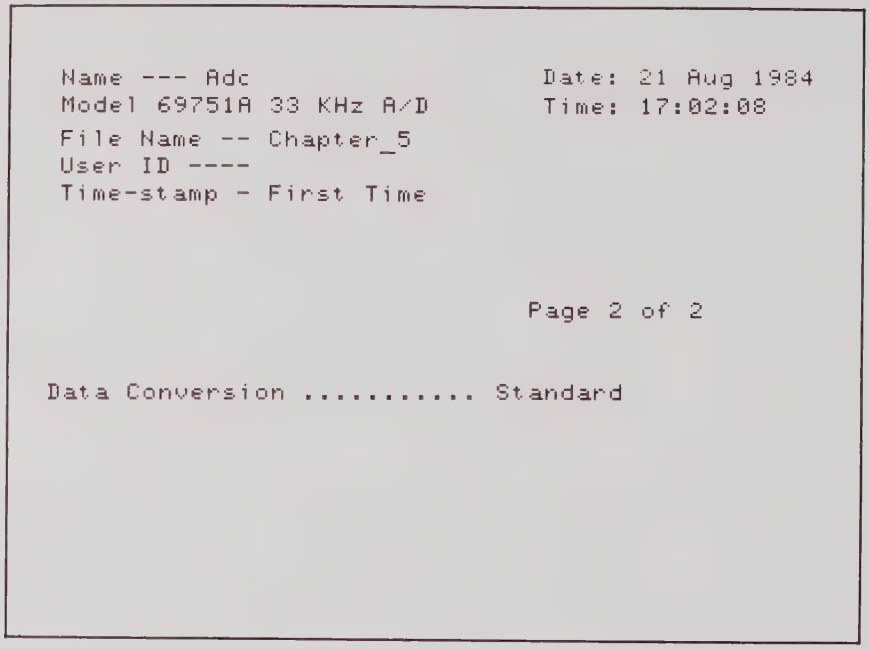
The analog-to-digital converter of the 69751A divides the span of the full-scale range into 4096 discrete steps. The LSB Value is the voltage change corresponding to one of those steps (least significant bit of a 12-bit answer). The LSB Values corresponding to the various ranges can be found in the 69751A Operating Manual. The default LSB Value is 0.005.

Scanner

Specifies whether or not the 69751A A/D card will reference a 69750A Scan Control card. If the 69751A does not have a front-end Scanner, then leave the default value of None. Otherwise, press the change key to step through each name listed in the COM statement and select the name of the Scanner menu or Scan Controller menu when it appears. If the name entered is not of either a Scanner menu or Scan Controller menu, an error will be generated. If the name entered has not yet been configured, an error will be generated. At this time, the user can continue, but the name **MUST** be configured before end menu. This is now a multiple card function which can be initialized via the Init statement using the name of the A/D card.

Timeout

Specifies the time in seconds that will be allowed for the A/D card to complete an analog-to-digital conversion. If an Input or Wait_for statement does not complete within this time period, Error Code 803 will be generated. The legal range is from 0.001 to 128,000 seconds. The default value is 5 seconds. Since the A/D card completes a conversion in 30 μs, the Timeout parameter is not meaningful if you program an A/D card with an Input statement, but is useful in conjunction with the External Trigger signal.



Menu Page 2 of 2

Data Conversion

Specifies how the data values that are returned from the A/D function in an Input or Get statement are to be scaled. One of three options can be selected: "Standard", "User", or "None".

1. "Standard" conversion is the default for the A/D function. The data values returned represent the standard units (volts) of the function.

2. "User" conversion allows the data values to be specified in terms of the application rather than in terms of the function's standard units (volts). As described below, four additional entries appear on the menu when User Conversion is selected.

User's
P1 P2

Standard
P1 P2

Appear on the menu only when User conversion is selected. These entries allow you to fill in the ratio of User and Standard values so that the 14752A can convert the data returned from the card into user defined units. The P1 default values are 0; the P2 default values are 1.

Menu Page 2 of 2 (continued)

Example: Assume that the A/D card is connected to a sensor that converts a pressure measured in pounds/sq. in. (psi) into a voltage: 0 psi is equal to 0 volts and 100 psi is equal to 10 volts. In order to program in "psi" instead of in "volts", select the "User" Data Conversion Option and enter the following P1 and P2 parameters.

	P1 Entry	**P2 Entry**
User's	0 psi	100 psi
Standard	0 volts	10 volts

3. "None" conversion specifies that the data returned from the A/D function represents the number of LSB's for the function. The range of values for this data is from −2048 to +2047 for a Bipolar range, or 0 to 4095 for a Unipolar range. For 69751A cards, as shipped from the factory, the values from −2048 to +2047 represent a bipolar input voltage range from −10.240 to +10.235 Volts (LSB = 0.005V).

Programming Fundamentals

Only two fundamental statements are needed to program the 69751A A/D Converter card. Each statement is described below and then a simple example program is given. For details concerning the syntax requirements for these statements, refer to Chapter 3.

Statement	Description
Init(Name)	Sets the card to the initial power-up state. The card's EOP and other control signals described in Chapter 10 are also cleared. Any previously enabled interrupts are disabled, and Timeout is reset to the initial value in the menu. If a scanner is specified, the 69750A Scan Controller card is loaded with the initial values specified in either the Scanner menu or the Scan Controller menu.
Input(Name,Value)	Initiates a data conversion in the A/D card and returns the reading into the variable "Value".

In the following example, the 69751A A/D card, identified by the Name "Volts_in", is programmed to take a single voltage reading. During the configuration phase, the Name is defined; the −10.240 to +10.235 V range and "Standard" Conversion are selected.

```
10 COM /Names/ Volts_in
20 !
30 Configure("Menu_on","ACDC")
40 !
50 ! Initialize the A/D.
60 !
70 Init(Volts_in)
80 !
90 ! Take a reading.
100 !
110 Input(Volts_in,A)
120 !
130 ! Display it.
140 !
150 DISP A
160 !
170 END
```

Advanced Programming

There are 14752A statements that allow special purpose programming of the 69751A. They are considered advanced because they require a deeper understanding of the interaction of the 14752A Programming Package with the 69751A. These statements are useful for external triggering and interrupts (see Chapter 10 for details). The advanced programming statements that can be used to program the 69751A A/D card are described below.

Statement	Description
Input_rblock(Name, Array(*)[,Count [,Index]])	Takes a block of readings from the A/D card, converts the data according to the specified data conversion, and returns the data into the REAL array called "Array". The number of words read from the card is equal to the size of the data array. The optional parameter Count,specifies the number of words that will be read from the card. This number must be less than, or equal to, the size of the array. Along with Count, Index can be used to specify the array location where the first data item will be placed. The data array **MUST** have been previously DIMensioned or ALLOCATEd in the user's program or a BASIC system error (#13) will be generated. Note that Input_rblock takes significantly longer than Input_iblock because of the extra data conversion time. If the A/D card is set for external triggering only, an external trigger **MUST** be supplied.
Input_iblock(Name, Iarray(*)[,Count[,Index]])	Takes a block of readings from the A/D card and stores the data into the INTEGER array called "Iarray". Since there is no data conversion, the Input_iblock takes a significant shorter amount of time than the Input_rblock. The number of words read from the card is equal to the size of the data array. The optional parameter COUNT, specifies the number of words that will be read from the card. This number **MUST** be less than, or equal to, the size of the array. Along with Count, Index can be used to specify the array location where the first data item will be placed. The data array must have been previously DIMensioned or ALLOCATEd or a BASIC system error (#13) will be generated. If the A/D card is set for external triggering only, an external trigger **MUST** be supplied.

NOTE

A Timeout error will occur if external triggers are not supplied within the time specified by the Timeout value. Also note that there is an overhead delay (5 ms. if Input_xblock immediately follows the start of the timebase) between the start of the external timebase and the Input_xblock statement that could cause a loss of data for some applications.

The Translate statement described below allows the user to separate the data conversion process from the measurement process. This capabiltiy is provided by combining this statement with Input_iblock. The Input_iblock statement is used to quickly collect raw integer data from the A/D card. The Translate statement can then be used to process this integer data into real numbers using the A/D card's specified data conversion.

Translate (Name, Iarray(*), User(*))

Used after Input_iblock to process the data of the A/D card. The A/D card is specified by the parameter, Name. The array named "Iarray" is an array of integers. This array is the same as "Iarray" used in Input_iblock, therefore, the array names are the same. The integer data is taken from Iarray and converted to real numbers. The data is then returned into the REAL array named "User". All arrays must have been previously DIMensioned or ALLOCATEd in the user's program otherwise a BASIC system error(#13) will be generated. An Input_iblock followed by a Translate statement is equivalent to an Input_rblock statement. The Translate statement, however, allows the user to delay the processing of data.

Start(Name1,...,Name8)

Initiates a voltage reading (conversion cycle) on a 69751A A/D card. All named functions (up to a total of 8) are "started" at the same time.

Get(Name,Value)

Returns a reading from the A/D card's storage register after it was placed there by a Start statement or an External Trigger signal.

Wait_for(Name)

Suspends program execution until the A/D card completes a conversion or until the timeout period has elapsed. Used if you want to wait for the 69751A to finish when it has been started with a Start statement or an External Trigger signal. Error code 803 is generated if timeout occurs before the card completes the conversion.

Check_done(Name,Flag)

Inquires about the status of the function's operation. Flag is set to a "1" if the conversion cycle has been completed. Once a value of one is returned, this statement will subsequently return a zero until the card has completed another conversion.

Enable_intr(Name,Vector)

Allows the A/D function to signal the computer via the interrupt system when it has completed a conversion. The "Vector" variable identifies the function that caused the interrupt. The Enable_intr statement must be used in conjunction with the interrupt statements described in Chapter 10.

Disable_intr(Name)

Inhibits the function from interrupting the computer. It is meaningful only after an Enable_intr statement has been sent. Note that the Init and System_init statements will also cancel any previously issued Enable_intr statements.

Set_timeout(Name,Value)

Changes the value of timeout from the time specified in the menu to Value. The legal range for Value is from 0.001 to 128,000 seconds.

69759A Analog-To-Digital Converter

The 69759A Analog-to-Digital Converter card measures voltage with a maximum conversion period of 2 μs. It employs the successive approximation technique. This means that the analog voltage is sampled during a narrow window (1 ns), and then converted to a 12-bit binary number.

This card can take measurements using one of 12 actual voltage ranges. Three full-scale ranges can be changed via on-card jumpers and four associated ranges can be selected via the menus or program statements. These ranges can also be selected by edge connector signals. If the range is selected internally, via the menus, the value is programmed to the card by the Init statement. To accomodate an application where several different ranges are desirable, multiple naming is convenient (see Chapter 10 for details). Instructions for modifying and recalibrating the card for other bipolar and unipolar ranges are given in Appendix A of this Manual, and complete hardware details are given in the 69759A Operating Manual (HP Part No. 69759-90001).

This card provides tremendous flexibility that can be controlled by program statements or edge connector signals. The user can disable or enable the internal and external triggers, and control the polarity of the external triggers.

Multiplexed A/D

The tri-state output feature of this card allows up to eight A/D cards to be multiplexed to one 69791A Memory Card System. This multiple card function can be configured via the Buffer menu discussed in Chapter 9. A complete example is given in Chapter 10 of this manual and Application Note 316-5, Data Capture.

DMA Transfers

The 69759A A/D Converter card utilizes the 6944A backplane signal line enhancements which allow Direct Memory Access (DMA). This capability provides high speed data transfers from the A/D card directly into computer memory at a maximum rate of 180 kHz for 64 k readings. This is initiated via an Input_iblock or Input_rblock statement when a 98620B DMA Controller card is present in the computer. If no DMA Controller card is present, data is transferred at a rate of 9 kHz.

Menu Entries for 69759A A/D Converter

The configuration and initial conditions for the 69759A A/D card are specified on a three page menu. Before programming the function, you **MUST** specify various parameters in the menu. The 69759A A/D function's parameters are described as follows:

```
Name --- Adc                          Date: 21 Aug 1984
Model 69759A 500 KHz A/D              Time: 16:31:16
File Name -- Chapter_5
User ID ----
Time-stamp - First Time

                                  Page 1 of 3
Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot ..........
Full-scale Range ......... +-10.24 volts
Initial Internal Range ... 10
Range Source .............. Internal
Scanner ................... None
Timeout ................... 5
```

Menu Page 1 of 3

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ...... 8
HP-IB Address .......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the 69759A A/D card. It has a legal range from 0 to 7. The default value is 0.

Slot

Identifies the location of the 69759A A/D card within the specified frame. It has a legal range from 0 to 15. The parameter has no default. It **MUST** be filled in.

Menu Page 1 of 3 (continued)

Full-scale Range

Specifies one of three jumper selectable input voltage ranges. The three possibilities are ±10.24, ±5.12 and 0 to 10.24. The default is ±10.24. Remember, to change this default value, on-card jumpers must be changed. Hardware details are given in the 69759A Operating Manual (HP Part No. 69759-90001).

Initial Internal Range

Specifies the desired nominal internal input range that will be programmed to the card via the "Init" statement. The choices appear according to the full-scale range chosen as shown in the following chart. For example, A full-scale range of ±10.24 corresponds to the nominal ranges of 10, 1,.1, and 100. The default value is 10.

The following chart specifies the 12 actual voltage ranges associated with each initial internal (nominal) range for the three fullscale ranges:

Full-scale Range (Jumper Setting)	Initial Internal Range (Programmable)	Actual Voltage Ranges
±10.24*	10*	−10.24 to 10.235*
	1	−1.024 to 1.0235
	.1	−0.1024 to 0.10235
	100	−102.4 to 102.35
±5.12	5	−5.12 to 5.1175
	.5	−0.512 to 0.51175
	.05	−0.0512 to 0.051175
	50	−51.2 to 51.175
0 to 10.24	10	0 to 10.235
	1	0 to 1.0235
	.1	0 to 0.10235
	100	0 to 102.35

*Default value

Range Source

Specifies whether the range is supplied internally, or externally at the card edge. If the range is supplied internally, it can be specified by a menu selection or a progam statement such as Set_range. The default is Internal.

Menu Page 1 of 3 (continued)

Scanner

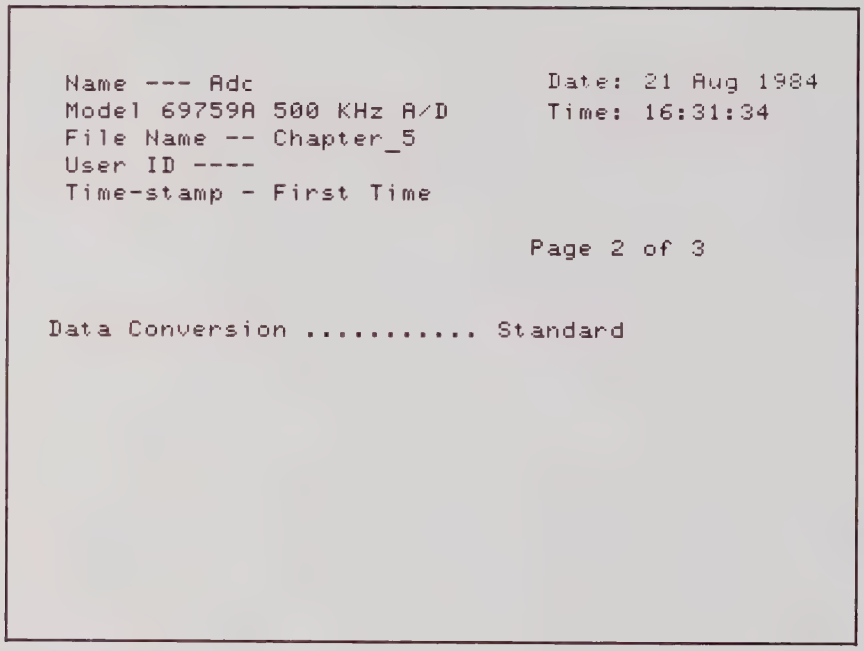
Specifies whether or not the 69759A A/D card will reference a 69750A Scan Control card. If the 69759A does not have a front-end Scanner, then leave the default value of None. Otherwise, press the change key to step through each name listed in the COM statement and select the name of either of the Scan Control card menus; i.e., Scanner or Scan Controller as described further in this chapter. If the name entered is not of either a Scanner menu or Scan Controller menu, an error will be generated. If the name entered has not yet been configured,an error will be generated. At this time, the user can continue, but the name **MUST** be configured before end menu. This is now a multiple card function which can be initialized via the Init statement using the name of the A/D card.

Timeout

Specifies the time in seconds that will be allowed for the A/D card to complete an analog-to-digital conversion. If an Input, Wait_for, Input_iblock or Input_rblock statement does not complete within this time period, Error Code 803 will be generated. The legal range is from 0.001 to 128,000 seconds. The default value is 5 seconds. Since the A/D card completes a conversion in approximately 2μs, the Timeout parameter is not meaningful if you program an A/D card with an Input statement. It is useful, however, in conjunction with the External Trigger signal and the Start statement when the internal trigger is disabled and the Gate signal is used to trigger an external timebase. This will be discussed in more detail in the Advanced Programming statement section.

NOTE

For Block transfers, using DMA (direct memory access), the timeout value depends on the speed of the processor. Because Series 300 computers operate at clock frequencies greater than 8 MHz, for these computers the actual timeout will be less than that specified by the programmed value. Increase the timeout value to compensate for the computer's faster speed.



Menu Page 2 of 3

Data Conversion

Specifies how the data values that are returned from the A/D function in an Input, Get or Input__rblock are to be scaled. One of three options can be selected: "Standard", "User",or "None".

1. "Standard" conversion is the default for the A/D function. The data values returned represent the standard units(volts) of the function.

2. "User" conversion allows the data values to be specified in terms of the application rather than in terms of the function's standard units (volts). As described below, four additional entries appear on the menu when User Conversion is selected.

User's
P1 P2

Standard
P1 P2

Appear on the menu only when User conversion is selected These entries allow you to fill in the ratio of User and Standard values so that the 14752A can convert the data returned from the card into user defined units. The default values are 0; the default values are 1.

Example: Assume that the A/D card is connected to a sensor that converts a flow rate measured in gallons per minute (gals/min) into voltage: 0 gals/min is equal to 0 volts and 50 gals/min is equal to 10 volts. In order to program in "gals/min" instead of in "volts", select the "User" Data Conversion Option and enter the following P1 and P2 parameters.

	P1 Entry	**P2 Entry**
User's	0 gals/min	50 gals/min
Standard	0 volts	10 volts

3. "None" conversion specifies that the data returned from the A/D function represents the number of LSB's for the function. The range of values for this data is from −2048 to +2047 for a Bipolar range, or 0 to 4095 for a Unipolar range. For 69759A cards, as shipped from the factory, the values from −2048 to +2047 represent a bipolar input voltage range from −10.240 to +10.235 Volts (LSB = 0.005V).

```
Name --- Adc                          Date: 21 Aug 1984
Model 69759A 500 KHz A/D              Time: 16:32:12
File Name -- Chapter_5
User ID ----
Time-stamp - First Time

                                  Page 3 of 3

Internal Trigger ..... Enabled
External Trigger ..... Disabled
Trigger Mode ......... Multiple
Trigger Polarity ..... Negative
Lockout Polarity ..... Disabled
Master Output Enable ..... Enabled
Mux Output Control ....... Disabled
External Output Enable .... Disabled
Gate Mode................. Multiple
Return Data With ..... Sign Extension
```

Menu Page 3 of 3

Internal Trigger

When enabled, the A/D card will respond to a Start statement. This is a programmatic (software generated) way of triggering the A/D card. When disabled, the Start statement is prevented from triggering the A/D card (see Start statement description). The default is Enabled.

External Trigger

When enabled, the A/D card will respond to the External Trigger signal (EXT). The external trigger can be generated from an external timebase, such as the Timer function described in Chapter 9, for pacing. When disabled external triggering is locked out. The default is Disabled.

Trigger Mode

Specifies whether the A/D card will respond to multiple external triggers (normal trigger mode) or to a single external trigger. With the single trigger mode, the A/D card will only respond to the first trigger that occurs. The Set_sngl_trig statement **MUST** be executed to allow the card to respond to each subsequent trigger. The default is Multiple.

Trigger Polarity

Trigger polarity can be selected as negative edge sensitive, positive edge sensitive, or external. A selection of negative indicates that the A/D card will be triggered on the negative edge of the External Trigger signal. A selection of positive indicates that the A/D card will be triggered on the positive edge of the External Trigger signal. An external selection specifies that the polarity of the external trigger signal is selected externally from the card edge. The default is negative.

Menu Page 3 of 3 (continued)

Lockout Polarity

The Lockout Polarity can be selected as disabled, positive, or negative. A selection of disabled indicates that the External Trigger Lockout signal is disabled. A selection of positive indicates that when the External Trigger Lockout signal is high,external triggers are locked out. A negative selection indicates that when the External Trigger Lockout signal is low, external triggers are locked out. The default is Disabled.

Master Output Enable

If enabled, either the Mux Output Control or the External Output Enable can disable the outputs. If disabled, the data outputs are disabled independent of the Mux Output Control and the External Output Enable (Also see Enable__mxctrl and Enable__een statement descriptions). The default is Enabled. Note this must be enabled for multiplexed A/D applications.

Mux Output Control

If enabled, the multiplex control circuitry can disable the data outputs. This is necessary with a multiplexed card configuration. If disabled, the multiplex control circuitry is prevented from disabling the data outputs. The default is Disabled. Note that this **MUST** be enabled for Multiplexed A/D applications.

External Output Enable

If enabled, the External Enable (EEN) signal Enable can disable the data outputs. If disabled, EEN is prevented from disabling the data outputs. The default is Disabled. Note that this should be disabled for multiplexed A/D application.

Gate Mode

In single gate mode, the Start statement causes the Gate signal to go active and remain so. This allows the A/D conversions to be paced by an external timebase. In multiple gate mode the Start, Input__iblock, or Input__rblock statement causes the Gate signal to go active until the A/D cards End of Conversion (EOC) signal resets the Gate. The default Multiple.

Return Data With

Specifies whether data is returned sign extended or with range information. Sign extension is useful when doing an Input__iblock with a known fixed range. Data must be returned with range information when the externally supplied range is unknown. The choices for this menu selection are sign extension and range readback. The default is Sign Extension.

Programming Fundamentals

Only two fundamental statements are needed to program the 69759A A/D Converter card. Each statement is described below and then a simple example program is given. For details concerning the syntax requirements for these statements refer to Chapter 3.

Statement	Description
Init(Name)	Sets the card to the values specified in the menus. The card's EOP and other control signals described in Chapter 10 are also cleared. Any previously enabled interrupts are disabled and Timeout is reset to the initial value in the menu. The card's initial modes and range are set to the values specified in the menu. If a Scanner is specified the 69750A Scan Control card is loaded with the initial values specified in either of the Scan Control card menus; i.e., Scanner or Scan Controller, as described in the associated sections of this chapter. The name parameter refers to the name of the A/D card.
Input(Name, Value)	Initiates a data conversion in the A/D card by setting range read-back. After the conversion completes, the reading is returned into the variable "Value".

In the following example, one 69759A A/D card, identified by the Name "Volts_in", is programmed to take a single voltage reading. During the configuration phase, the Name is defined; the ± 10.24 V range , the 10 V Initial Internal Range and the "Standard " Conversion are selected.

```
10  COM/Names/Volts_in
20  !
30  Configure("Menu_on","ACDC")
40  !
50  ! Initialize the A/D
60  !
70  Init(Volts_in)
80  !
90  ! Take a reading.
100 !
110 Input(Volts_in,A)
120 !
130 ! Display it.
140 !
150 DISP A
160 !
170 END
```

Advanced Programming

There are many 14752A statements that allow special purpose programming of the 69759A. They are considered advanced because they require a deeper understanding of the interaction of the 14752A. Programming Package with the 69759A. These advanced statements provide the user with the capability of controlling many card edge input signals, such as the External Trigger, the External Trigger Lockout and the External Enable. Other statements are useful for such advanced concepts as external triggering, interrupts, and multiplexing. A multiplexed configuration is a combination of up to eight 69759A A/D cards with one 69791A Memory Card System. This multiple card function is definable through the Buffer menu (see Chapter 9 details). The advanced programming statements that can be used to program the 69759A A/D card are described below.

Statement	Description
Start(Name1,...,Name8)	Generates a Gate signal which will initiate a voltage reading on the A/D card if the internal trigger is enabled. All named functions are started at the same time. If the internal trigger is disabled, a Gate signal is still generated but the A/D card is not directly triggered. This could be used to trigger an external timebase to pace the A/D conversions.
Get(Name,Value)	Returns a reading from the A/D card's storage register after it was placed there by a Start statement or an external trigger. If the range is set to external and the range readback is not set, the range information is not available to provide data conversion.In this case, if a Get statement is executed with a data conversion of either Standard or User, Error code 846 will result.
Enable_intr(Name,Vector)	Allows the A/D card to signal to the computer via the interrupt system when it has completed a conversion. The "Vector" variable identifies the function that caused the interrupt. The Enable_intr statement must be used in conjunction with the interrupt statements described in Chapter 10.
Disable_intr(Name)	Inhibits the A/D card from interrupting the computer. It is meaningful only after an Enable_intr statement has been sent. Note that the Init and System_init will also cancel any previously issued Enable_intr statements.
Check_done(Name,Flag)	Inquires about the status of the function's operation. Flag is set to a "1" if the conversion cycle has been completed. Once a value of "1" is returned, this statement will subsequently return a "0" until the card has completed another conversion.
Wait_for(Name)	Suspends program execution until the A/D card completes a conversion or until the timeout period has elapsed. Used if you want to wait for the A/D card when it has been started with a Start statement or an external trigger signal. Error code 803 is generated if timeout occurs before the card completes the conversion.

Advanced Programming (continued)

Statement	Description
Input__rblock(Name, Iarray(*)[,Count[,Index]])	Takes a block of readings from the A/D card, converts the data according to the specified data conversion, and returns the data into the REAL array called "Array". The number of words read from the card is equal to the size of the data array. The optional parameter Count, specifies the number of words that will be read from the card. This number must be less than, or equal to, the size of the array. Along with Count, Index can be used to specify the array location where the first data item will be placed. The data array must be previously DIMemsioned or ALLOCATEd in the user's program or a BASIC system error (#13) will be generated. The speed of the block transfer depends upon whether or not there is a DMA card present in the controller. The transfer rate is 180 kHz with the DMA card and 10 kHz without it. If there is a DMA card present in the system and a channel is available, the A/D card and Multiprogrammer Interface card will be programmed for DMA. Either the Set__int__xfer or Set__ext__xfer command (or their associated menu selections) **MUST** be initiated before this Input__rblock statement. If the range of the A/D card has been set to external, the range information must be held until End of Conversion (EOC). Note: the Input__rblock takes significantly longer than the Input__iblock because of the extra data conversion time.

NOTE

If your computer has cache memory, you may get a BASIC error (388) when using this statement with DMA transfers. In this case, turn off cache memory before executing the statement and turn it on again after the statement. The command to turn cache memory off is:

CONTROL 32,1;0

The command to turn cache memory on is:

CONTROL 32,1;1

Input_iblock(Name, Iarray(*)[,Count[,Index]])

Takes a block of readings from the A/D card and stores the data into the INTEGER array called "Iarray". Since there is no data conversion, the Input_iblock takes significantly less time than the Input_rblock(see Set_range_rdbk and Set_sign_extn statements). The number of words read from the card is equal to the size of the data array. The optional parameter Count, specifies the number of words that will be read from the card. This number must be less than or equal to the size of the array. Along with Count, Index can be used to specify the array location where the first data item will be placed. The data array must have been previously DIMensioned or ALLOCATEd in the user's program or a BASIC system error (#13) will be generated. The speed of the block transfer depends upon whether there is a DMA card present or not. The transfer rate is 180 kHz with the DMA card and 9 kHz without it. If there is a DMA card present in the system and a channel is available, the A/D card and Multiprogrammer Interface card will be programmed for DMA. Either the Set_int_xfer or Set_ext_xfer command (or their associated menu selections) **MUST** be initiated before this Input_iblock statement to select internal or external pacing.

NOTE

If your computer has cache memory, you may get a BASIC error (388) when using this statement with DMA transfers. In this case, turn off cache memory before executing the statement and turn it on again after the statement. The command to turn cache memory off is:

CONTROL 32,1;0

The command to turn cache memory on is:

CONTROL 32,1;1

Advanced Programming (continued)

The Translate statement described below allows the user to separate the data conversion process from the measurement process. This capability is provided by combining this statement with Input__iblock. The Input__iblock statement is used to quickly collect raw integer data from the A/D card. The Translate statement can then be used to process this integer data into real numbers using the A/D card's specified data conversion.

Statement	Description
Translate(Name,Iarray(*), User(*)[,Range(*)])	Used after Input__iblock to process the data of the A/D card. The A/D card is specified by the parameter, Name. The array named "Iarray" is an array of integers. This is the same array as "Iarray" used in Input__iblock, therefore, the names are the same. The integer data is taken from Iarray and converted to real numbers. The data is then returned into the REAL array named "User". The optional range array parameter is used to process the results when the range source of the A/D is set to external. Depending upon the full-scale range selected, the range array **MUST** contain either 10,1,.1,100 values, or 5,.5,.05,50 values. Each value in the range array will be associated with the corresponding value in Iarray. If the range array is smaller than Iarray, the ranges will be reused until Iarray is exhausted. For example, if the range array contains five elements and Iarray contains 15 elements, each range value will be used three times (range value #1 will be used with Iarray value #1, 6 and 11, value #2 will be used with Iarray value #2, 7 and 12, etc.). This range array can be the same array used in the Load__channels statement. All arrays **MUST** be previously DIMensioned or ALLOCATEd in the user's program otherwise a BASIC system error (#13) will be generated. An Input__iblock followed by a Translate statement is equivalent to an Input__rblock statement. The Translate statement, however, allows the user to delay the processing of data.
Set__timeout(Name,Value)	Changes the value of timeout from the time specified in the menu to "Value". The legal range for Value is from 0.001 to 128,000 seconds.
Set__range(Name,Range)	Sets the internal input range to the value specified by the parameter "Range". The specified range must be within 2.5% of one of the nominal input ranges (refer to Initial Internal ranges in the previous table) or Error code 847 will be generated.
Set__range__int(Name [,Range])	Allows the range to be supplied internally. If the optional Range parameter is not specified the range value specified by the previous Set__range command remains unchanged. Otherwise, the range must meet the requirements as specified in the above Set__range command.

Advanced Programming (continued)

Statement	Description
Set__range__ext(Name)	Allows the range to be supplied externally at the card edge. A Set__range command has no effect on the range value in this case. If a Set__range command is initiated, and the range selection is external, the range value is saved until a Set__range__int is executed.
Enable__intrig(Name)	Allows the A/D card to respond to internal triggers. A Start command will cause the A/D card to take a reading.
Disable__intrig(Name)	Prohibits A/D card from responding to internal triggers. A Start command is prevent from triggering the A/D card.
Enable__extrig(Name)	Allows the A/D card to respond to external triggers at the card edge. An external trigger will cause the A/D card to take a reading.
Disable__extrig(Name)	Prohibits the A/D card from responding to external triggers at the card edge.
Enable__ext__tps(Name)	Allows the polarity of the External Trigger signal to be supplied externally at the card edge.
Set__extrig__lo(Name)	Allows the polarity of the External Trigger signal to be supplied internally and sets the External Trigger signal to be active low.
Set__extrig__hi(Name)	Allows the polarity of the external trigger signal to be supplied internally and sets the external trigger signal to be active high.
Set__extlock__lo(Name)	Allows the External Trigger Lockout signal to prevent external triggering when the External Trigger Lockout signal is low.
Set__extlock__hi(Name)	Allows the External Trigger Lockout signal to prevent external triggering when the External Trigger Lockout signal is high.
Disable__extlock(Name)	Disables the external trigger lockout signal.
Set__single__trig(Name)	Allows the A/D card to respond only to the first trigger (internal/external) that occurs. To respond to subsequent single triggers, this statement needs to be re-executed each time.
Set__multi__trig(Name)	Allows the A/D card to respond to multiple triggers (internal/external). This returns the card to normal trigger mode after it has been set to single.
Enable__mxctrl(Name)	Allows the CHAIN IN and CHAIN OUT signals of the multiplex control circuitry to disable the data outputs.These signals determine which A/D card will transfer data to the 69791A Memory Card System in a multiplexed application. The outputs must be enabled (by the

Advanced Programming (continued)

Enable__output statement or the Master Output Enable menu selection) and EEN must be true or disabled (by the Enable__een statement or the External Output Enable menu selection).

Disable__mxctrl(Name) Prohibits the multiplex control circuitry from disabling the data outputs.

Enable__output(Name) Allows either the multiplex control circuitry or the External Enable (EEN) signal to disable the data outputs.

Disable__output(Name) The data outputs are disabled regardless of the multiplex control circuitry or the External Enable signal(EEN).

Enable__een(Name) Allows the External Enable signal to disable the data outputs. The outputs must have been previously enabled (by an Enable__output statement or by the Master Output Enable menu selection).

Disable__een(Name) Prohibits the External Enable signal from disabling the data outputs.

Set__single__gate(Name) When in single gate mode, a Start, Input__iblock, or Input__rblock causes the gate signal to go active and remain so. This signal is used to trigger an external timebase (such as the Timer function discussed in Chapter 9) to pace the A/D conversions. To return the gate signal back to its inactive position, re-execute the Set__single__gate statement.

Set__multi__gate(Name) When in multiple gate mode, a Start, Input__iblock, or Input__rblock causes the gate signal to go active until it is reset by the A/D card's End of Conversion (EOC) signal.

Set__sign__extn(Name) Allows data to be returned sign extended. This provides a 16 bit integer from a 12 bit A/D card reading. Sign extension is useful when doing an Input__iblock directly from the A/D card to computer memory with a fixed range.

Set__range__rdbk(Name) Specifies that the returned data will have range information. This **MUST** be set when using either the Get statement or the Input__iblock with externally supplied range. Note that if the range is set to external, the range information must be held until End of Conversion (EOC).

Set__ext__xfer(Name) Prepares the A/D card for an Input__iblock or Input__rblock with external pacing. To achieve the equivalent capability via the menus, the following selections must be made:

Internal Trigger	Disabled
External Trigger	Enabled
Gate Mode	Single
Trigger Mode	Multiple

Advanced Programming (continued)

Statement	Description
Set__int__xfer(Name)	Prepares the A/D card for an Input__iblock or Input__rblock with no external pacing. To achieve the equivalent capability via the menus, the following selections must be made: Internal Trigger .....Enabled External Trigger .....Disabled Gate Mode.....Multiple Trigger Mode .....Multiple

Scanning A/D Programming Example

The following advanced programming example illustrates how to use the 69759A A/D card, together with the 69750A Scan Control card (and 69752A FET Scanner card) to form a Scanning A/D multiple card function. The card interconnections for this example are shown below and the menu configurations and Program listing follow.

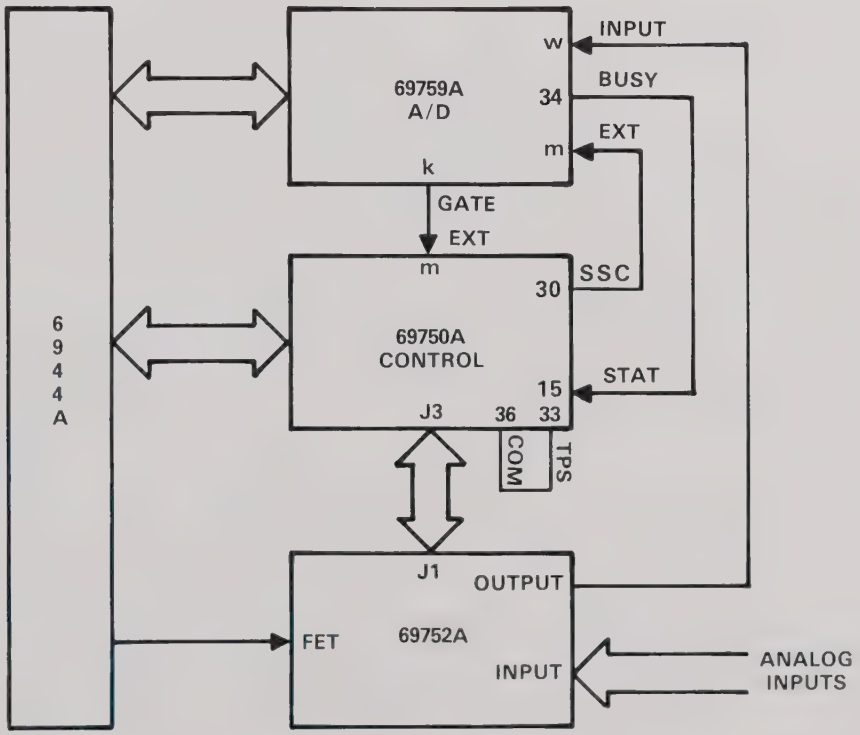


Figure 5-1. Scanning A/D Card Interconnections*

*Do not connect the Scanner isolated common to anything. Use the A/D input signal common instead.

Menu Configurations for 69759A

```
Name --- Adc                          Date: 21 Aug 1984
Model 69759A 500 KHz A/D              Time: 17:13:01
File Name -- Chapter_5
User ID ----
Time-stamp - First Time

                                    Page 1 of 3
Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot .......... 1
Full-scale Range  ........ +-10.24 volts
Initial Internal Range ... 10

Range Source ............. Internal
Scanner .................. Seq_scan
Timeout .................. 5
```

Menu Page 1 of 3

```
Name --- Adc                          Date: 21 Aug 1984
Model 69759A 500 KHz A/D              Time: 17:13:19
File Name -- Chapter_5
User ID ----
Time-stamp - First Time

                                    Page 2 of 3

Data Conversion ........... Standard
```

Menu Page 2 of 3

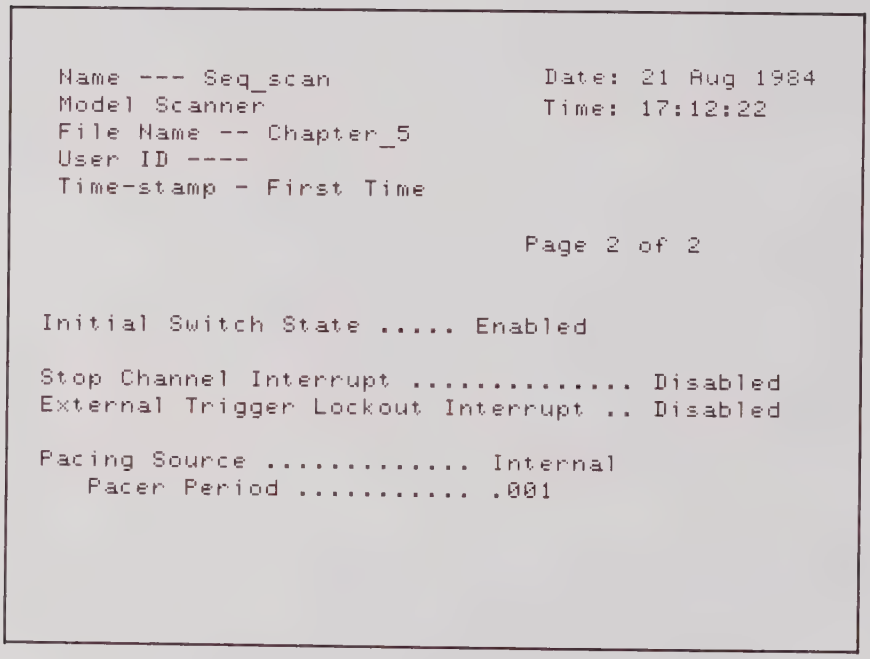
```
Name --- Adc                          Date: 21 Aug 1984
Model 69759A 500 KHz A/D              Time: 17:13:58
File Name -- Chapter_5
User ID ----
Time-stamp - First Time

                                  Page 3 of 3

Internal Trigger ..... Disabled
External Trigger ..... Enabled
Trigger Mode ......... Multiple
Trigger Polarity ..... Negative
Lockout Polarity ..... Disabled
Master Output Enable ..... Enabled
Mux Output Control ....... Disabled
External Output Enable .... Disabled
Gate Mode................. Single
Return Data With ..... Sign Extension
```

Menu Page 3 of 3

Menu Configurations for Scanner

```
Name --- Seq_scan                     Date: 21 Aug 1984
Model Scanner                         Time: 17:11:37
File Name -- Chapter_5
User ID ----
Time-stamp - First Time

                                  Page 1 of 2

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot .......... 5
Scanner Mode ..... Sequential
    Start Channel ..... 1
    Stop Channel ....... 10

Timeout ....... 5
```

Menu Page 1 of 2



Menu Page 2 of 2

In the following example, 10 channels are sequentially scanned every minute for 10 minutes. The channel-to-channel period is 1 ms (1E-3). This is an illustration of a Scanning A/D multiple card function where the 69759A A/D card in slot #1 references the 69750A Scan Control card in slot #5 via the Scanner menu (see Scanner menu further in this chapter). The 69759A A/D Converter card is identified by the name Adc, and the 69750A Scan Control card is identified by the name Seq__scan.

```
10     COM /Names/ Adc,Seq_scan
20     DIM Readings(0:99)
30     Configure("Menu_on","Seq_config")
40     Minutes=0
50     ON CYCLE 60 GOSUB Take_a_reading
60 Continue:IF Minutes<>10 THEN Continue
70     ! Otherwise stop
80     OFF CYCLE
90     ! Print all readings after 10 minutes
100    PRINT Readings(*)
110    GOTO Done
120    !
130    ! Initialize the A/D card and the Scan Control card
140    ! with the values specified in the menus each time
150    ! readings are taken.
160    !
170 Take_a_reading:Init(Adc)
180    Index=10*Minutes
190    !
200    ! Initialize a Gate signal from the A/D card
210    ! which triggers the Scan Control card and
220    ! begins the internal pacer. Scan 10 specified
230    ! channels and store the resultant readings in
240    ! Readings array.
250    !
260    Input_rblock(Adc,Readings(*),10,Index)
270    Stop(Seq_scan)
280    Minutes=Minutes+1
290    RETURN
300 Done:END
```

69750A Scan Controller

The 69750A Scan Control card and one or more of the Scanner cards, 69752A, 69754A, or 69755A, provide a general purpose Scanner implemented with either FET switches (69752A or 69755A) or dry reed relays (69754A). The FET switches have fast settling times (less than 10 μs) and can handle single ended input signals of $\pm$ 10.24 volts. The dry reed relay switches will close in 1 ms and handle either single-ended or double-ended input signals up to $\pm$ 100 volts.

The 69752A, 69754A, and 69755A cards are not directly programmable, and only operate under control of the 69750A. The 14752A CAT Programming Package interacts with the 69750A only, not with the Scanner cards. The 69750A communicates with the Scanner card(s) via chaining cables that interconnect the cards.

NOTE

For EMI reasons, a Scanner System should be installed in a single Multiprogrammer unit. Since a unit has 16 slots, one 69750A card and up to 15 Scanner cards can be installed. Thus, if fifteen 69752A 64-channel FET Scanner cards are installed, the maximum channel count is 960 (15X64) even though there are 1024 addresses available.

Refer to the Operating Manuals for the 69750A Scan Control card and the applicable Scanner Card (69752A, 69754A, or 69755A) for complete specifications and hardware details. The HP Part No's. for these manuals are: 69750-90001, 69752-90001 (covers the 69752A and 69755A), and 69754-90001. Detailed information on operations and applications of the Scanner cards are provided in Application Note 316-3, "High Speed FET Scanning", and detailed programming information is provided in the Application Note 316-5 on Data Capture.

The 69751A or 69759A A/D Converter card can be combined with the 69750A Scan Control card and one or more Scanner cards, to allow measurement of many different input signals. Channel to channel pacing is provided by the 69750A Scan Control card. This combination of the 69751A or 69759A A/D converter with the 69750A Scan Control card and one or more Scanner cards is known as a multiple card function. In general, a multiple card function is a combination of two or more cards working together as a "subsystem".

Once the cards have been linked together in hardware, they need to be initialized (via the Init statement) as a subsystem in preparation for an analog measurement. The software provides a means to do this. In order to provide such features, the user must specify how the cards are cabled together by supplying the name of either one of the associated menus, the Scanner menu (as explained in the next section of this chapter), or the Scan Controller menu, in the 69759A or 69751A A/D card

menu. This establishes a front-end link between the A/D card and the Scan Control card. This means that a single Init statement using the A/D card name, such as Init(Adc), will initialize all of the cards in the multiple card configuration. This is, however, the only 14752A statement that will act upon the entire subsystem. All other 14752A statements will act upon the individual cards.

If so desired, a Memory function (69790B/69791A) can be added to this multiple card function to allow measurement and local storage of many different input signals. The local storage buffer (69790B Memory card or 69791A Memory Card System) frees the computer from the task of reading data directly from the A/D card and allows the function to operate at a rate that is independent of the computer's speed. This function is configured via the Buffer menu (see Chapter 9 for details).

The Scan Controller menu should be used for general purpose applications where all of the features of the Scan Control card are required. The Scan Controller menu provides Crosspoint Switch Mode selection.The configuration and initial conditions for the 69750A Scan Controller menu are specified on the three page Scan Controller menu. You are required to fill in the various parameters of the menu in order to program the function.

Menu Entries for 69750A Scan Controller

The Scan Controller function's parameters are described as follows:

```
  Name --- Scan_control          Date: 21 Aug 1984
  Model 69750A Scan Controller   Time: 17:03:59
  File Name -- Chapter_5
  User ID ----
  Time-stamp - First Time

                                 Page 1 of 3

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot ..........



Timeout ....... 5
```

Menu Page 1 of 3

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ..... 8
HP-IB Address ......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the 69750A Scan Control card. It has a legal range from 0 to 7. The default value is 0.

Menu Page 1 of 3 (continued)

Slot

Identifies the location of the 69750A Scan Control card within the specified frame. It has a legal range from 0 to 15. (The slot address of any Scanner cards is not required.) This parameter has no default value; it **MUST** be filled in.

Timeout

Specifies the time in seconds that will be allowed for the 69750A card to complete operation. The 69750A completes operation after it closes a switch or starts the on-board Pacer. If the card does not complete within the time specified for Timeout, an error message (Code 803) is generated.

```
Name --- Scan_control          Date: 21 Aug 1984
Model 69750A Scan Controller   Time: 17:04:24
File Name -- Chapter_5
User ID ----
Time-stamp - First Time

                                Page 2 of 3

Scanner Mode ..............  Sequential
Switch Mode ...............  Single Point
Channel Address Source ....  Internal
   Start Channel Address ..  0
   Stop Channel Address ...  15

Pacing Source .............  Internal
   Pacer Period ...........  4.E-5
```

Menu Page 2 of 3

Scanner Mode — Specifies the mode used to select channels, either Sequential (from start channel to stop channel) or Random. The default is Sequential.

Switch Mode — Specifies whether Singlepoint (only a single channel can be closed) or Crosspoint (more than one channel can be closed at one time) is in effect. The default is Singlepoint.

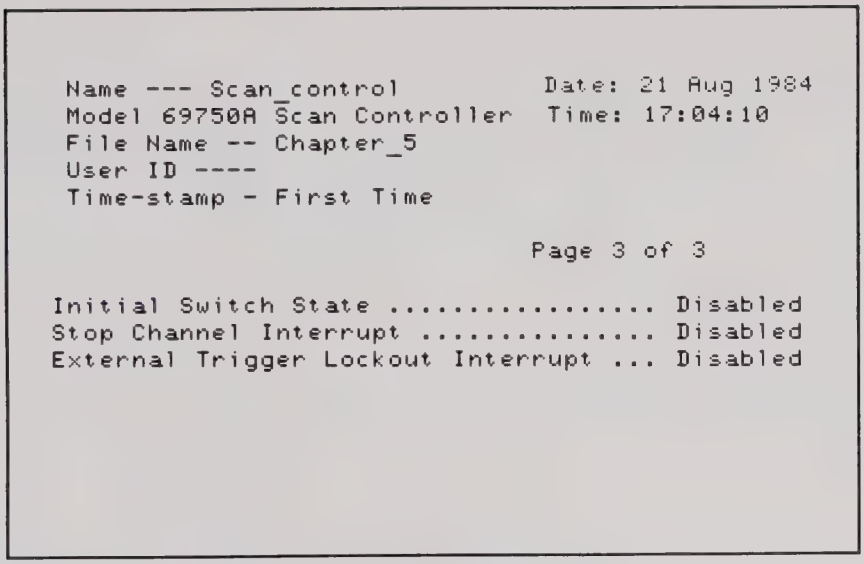
Channel Address Source — Channel addresses can either be supplied by the program (Internal) or taken from the external address inputs (IN00-IN09) on the card edge (External). The default, is Internal.

Start Channel Address — Appears only when "Internal" is selected as the Channel Address Source. For sequential scan, this parameter specifies the first channel in the range of channels to be scanned. For a random scan, this parameter specifies the first channel selected. The legal range is from 0 to 1023. The default is 0.

Stop Channel Address — For a sequential scan, this parameter specifies the last channel in the range of channels to be scanned. For either sequential or random scanning modes, the Scan Control card can be programmed to interrupt or indicate completion when the current channel is equal to the Stop Channel. The legal range is from 0 to 1023. The default is 15.

Pacing Source — The pacing source can be provided by an external signal on the card edge or by the internal Pacer on the Scan Control card. The default is Internal. Selecting Internal, brings up the next field, Pacer Period.

Pacer Period — Determines the scan rate for the internally paced scan. The period is specified in seconds from .000001 to 65.535. The default is .00004 seconds.


Name --- Scan_control
Date: 21 Aug 1984
Model 69750A Scan Controller
Time: 17:04:10
File Name -- Chapter_5
User ID ----
Time-stamp - First Time
Page 3 of 3
Initial Switch State ..... Disabled
Stop Channel Interrupt ..... Disabled
External Trigger Lockout Interrupt ... Disabled

Menu Page 3 of 3

Initial Switch State

Indicates whether the Scanner card switches will be Enabled or Disabled after execution of the Init statement. The default, Disable, holds all the switches open until execution of the Set__switches(Name,1) statement.

Stop Channel Interrupt

Specifies whether or not accessing the Stop Channel will interrupt the computer. Selecting "Enabled" allows an interrupt to be generated when the current channel address is equal to the Stop channel address. The Enable__intr statement must have been previously executed (see Enable__intr statement).

External Trigger Lockout Interrupt

Specifies whether or not the computer will be interrupted when the External Trigger Lockout condition occurs. Selecting "Enabled" allows an interrupt to be generated when the External Trigger Lockout is enabled via the Lockout input signal on the card edge or the execution of the Enable__lockout statement. The Enable__intr statement must have been previously executed (see Enable__intr statement).

Programming Fundamentals

Programming the 69750A Scan Control card via the Scan Controller menu is complex because the menu is provided for general purpose applications. This menu supports the maximum flexibilty that the Scan Control card was designed for. To use the Scan Controller menu first familiarize yourself with the operation of the card as described in the 69750A Operating Manual (HP Part No. 69750-90001, and then proceed to the Advanced Programming Section which follows. Note that for straight forward Scanning applications, the Scanner menu covered in the next section may be more appropriate to use.

Advanced Programming

The 14752A programming statements provided for the 69750A Scan Control Card are presented below, followed by an example program.

Statement	Description
Init(Name)	Sets up the initial configuration specified in the Scan Controller menu (Scanner Mode, Switch mode, Channel Address Source, Start Channel, Stop Channel, Pacing Source, Pacing Period, Initial Switch State, Initial Interrupt Mask and Timeout). Interrupts from the card are disabled as if a Disable_intr statement were executed.
Start(Name1,...,Name8)	For the Scan Controller menus the action of this statement depends on the last statement issued to the card. If the last statement sent to the card was a Set_period or a Stop statement, then the Start statement starts the on-board pacer. If the last statement was any other than the two mentioned, then the Start statement will cause a channel to be selected.
Enable_intr(Name,Vector)	Writes the current Interrupt Mask to the Scan Control card, enabling the specified interrupts. The parameter "Reason" in your interrupt service routine "User_handler" can be checked to determine the source of the 69750A interrupt. Note that a value of 3 for "Reasons" indicates a "do not know" case; both interrupts (to channel and external trigger interrupts) were enabled at the same time and the source could be either one. "Vector" identifies the function that caused the interrupt. This statement must be used with the interrupt statements explained in Chapter 10.
Disable_intr(Name)	Disables all interrupts on the card. The card Interrupt Mask is not affected.

Advanced Programming (continued)

Statement	Description
Check__done(Name,Flag)	Checks if card has completed operations. The 69750A card completes operations after it closes a switch or starts the pacer. If the card has completed, Flag is set to a value of "1". If the card has not completed, Flag is set to "0". Once a value of "1" is returned, subsequent calls to "Check__done" will return a "0" until the card completes another operation.
Wait__for(Name)	Suspends program execution until the card has completed operations or a Timeout has occured. Used if you want to wait for the card to finish when it was started either by an external signal or a Start statement.
Set__timeout(Name,Value)	Changes the value of timeout from the time specified in the menu to Value. The legal range for Value is from 0.001 to 128,000 seconds.
Set__switches(Name,Flag)	This statement controls the override feature built into the Scanner cards. If Set__switches(Name,0) is executed, any selected switches will be forced open. Selecting any further switches, will cause them to be remembered, but not closed. After the execution of the Set__switches(Name,1) statement, any presently selected switches will close. This feature is useful as a general purpose override or to synchronize the closing of several switches in Crosspoint Mode. Note that on a per-card basis (models 69752A, 69754A, or 69755A) only one switch can be closed at any time.
Enable__extrig(Name) **Disable__extrig(Name)**	These statements control the state of the external trigger lockout flip-flop on the Scan Control card. The external trigger input on the card edge can be used to perform the same operations as the Start statement. The external trigger lockout flip-flop can be used to mask off the effect of the External Trigger signal. Disable__extrig masks off the external trigger signal, while Enable__extrig allows it to pass through. (Note: there are hardware signals on the card edge, Lock and Unlock, which can override the action of these statements.)
Set__singlepoint(Name) **Set__crosspoint(Name)**	The normal operating mode of the Scan Control card, Singlepoint, allows no more than one switch on any of the Scanner cards (models 69752A, 69754A, and 69755A) to be closed at any time (closing a switch on a card opens all other switches on all cards). The Crosspoint Mode allows several switches on different Scanner cards to be closed at the same time; however, no more than one switch at a time can ever be closed on any one card. Use the applicable statement to select the desired operating mode.

Advanced Programming (continued)

Statement	Description
Set__stop__ien(Name,Flag)	This statement programs the bit in the card Interrupt Mask that controls the interrupt generated when the current channel address equals the Stop channel address. If Flag = 1, generation of this interrupt is allowed; if Flag = 0, not allowed. The mask does not take effect until the Enable__intr statement is executed. If this interrupt occurs, the value of "Reason" returned to your User__handler routine will be set to a "1".
Set__extlock__ien(Name, Flag)	This statement programs the bit in the card interrupt mask that controls the interrupt generated when the external trigger lockout is enabled. If Flag = 1, generation of this interrupt is allowed; if Flag = 0, not allowed. The mask does not take effect until the Enable__intr statement is executed. If this interrupt occurs, the value of "Reason" returned to your User__handler routine will be set to a "2".
Set__sequen__mode(Name)	Sets the mode of the Scan Control card to sequential. Channels will be selected sequentially from the Start channel address to the Stop channel address.
Set__random__mode(Name)	Sets the mode of the Scan Control card to random. Channel address will be provided by the program or via the external address inputs on the card edge (IN00-IN09).
Set__int__start(Name) **Set__ext__start(Name)**	These statements determine the source of channel addresses supplied to the Scan Control card. Channel addresses can be supplied by the program (internally) or via external address lines (IN00-IN09).
Set__int__pacing(Name) **Set__ext__pacing(Name)**	These statements determine the source of scan pacing. The Set__int__pacing statement selects the on-board pacer, while the Set__ext__pacing statement allows for channel advance via program control or the external trigger input.
Set__period(Name,Value)	Programs the period of the on-board pacer. The period can have values from 1 μs to 65.535 ms with 1 μs resolution, or from 66 ms to 65.535 seconds with 1 ms resolution.
Stop(Name)	Stops the pacer without altering the period. If the next statement sent to this card is a Start statement, the pacer will be started (as opposed to having the Start statement select a channel).
Write__start__chn(Name, Value)	Sets the channel address source to internal and selects the channel specified by Value. Used to specify the Start channel in the sequential mode. The legal range of channels is from 0 to 1023.

Advanced Programming (continued)

Statement	Description
Write_stop_chnl(Name, Value)	Sets the stop channel address to the value specified The legal range of channel addresses is from 0 to 1023.
Write_ext_chnl(Name)	Sets the channel address source to external and selects the channel via the external address lines (IN00-IN09). Used to select channels in the random mode.
Read_start_chnl(Name, Value)	Returns the Start Channel Address into the variable, "Value". The legal range of addresses is from 0 to 1023.
Read_chnl_addr(Name, Value)	Returns the Current Channel Address into the variable, "Value". The legal range of addresses is from 0 to 1023.
Read_stop_chnl(Name, Value)	Returns the Stop Channel Address into the variable "Value". The legal range of addresses is from 0 to 1023.
Set_timeout(Name,Value)	Changes the value of timeout from the time specified in the menu to Value. The legal range for Value is from 0.001 to 128,000 seconds.

Advanced Programming Example

The following program example illustrates how two 69754A Relay Scanner cards can be used with the 69750A Scan Control/Pacer card to select one of 32 inputs and route it to one of 32 outputs.

During the configuration phase, the 69750A Scan Controller is identifed by the name "Route" and the initial conditions are set. The initial conditions include selecting the Random scanning mode and Crosspoint switch mode. The Crosspoint mode is used to allow one channel on each card to be closed. The set__switch statements are used to ensure a "break-before-make" connection.

```
10 COM /Names/ Router
20 !
30 Configure("Menu__on","Test")
40 !
50 ! System__init
60 !
70 !Prompt for a source and destination channel.
80 !
90 Select__channel:!
100 INPUT "Enter a source and destination channel" A,B
110 Set__switches(Router,0)
120 Write__start__chn(Router,A)
130 Write__start__chn(Router,B)
140 Set__switches(Router,1)
150 GOTO Select__channel
160 !Loop indefinitely.
170 END
```

Scanner Function

The 69750A Scan Control card has two associated menus, Scanner and Scan Controller. The Scanner menu has been provided strictly for scanning applications where all of the features of the 69750A Scan Control card are not required. It is the menu that is most commonly used with multiple card functions involving the Scan Control card. For a multiple card function involving random scanning, this menu allows you to associate external channel memory with the scanning function. This feature provides a Load__channels program statement that will load the external channel address memory with the channels to be scanned.

The Scanner menu is used for sequential and random scanning. For this menu, the start statement and external trigger are not used for external channel selection. These statements are used to start the on-board pacer.

The configuration and initial conditions for the Scanner menu are specified on the following two page menu. You are required to fill in the various parameters of the menu in order to program the function. Complete information about the 69750A Scan Control card, the associated Scanner cards, and multiple card functions can be found in the 69750A Scan Controller section of this chapter. It is recommended that this be read before configuring the Scanner menu.

Menu Entries for Scanner

The Scanner function's parameters are described as follows:

```
Name --- Scanner                    Date: 21 Aug 1984
Model Scanner                       Time: 17:03:19
File Name -- Chapter_5
User ID ----
Time-stamp - First Time

                                 Page 1 of 2

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot ..........
Scanner Mode ..... Sequential
    Start Channel ..... 0
    Stop Channel ..... 15

Timeout ..... 5
```

Menu Page 1 of 2

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ..... 8
HP-IB Address ......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Menu Page 1 of 2 (continued)

Frame

Specifies the Multiprogrammer frame which contains the 69750A Scan Control card. It has a legal range from 0 to 7. The default value is 0.

Slot

Identifies the location of the 69750A Scan Control card within the specified frame. It has a legal range from 0 to 15. (The slot address of any Scanner cards is not required.) This parameter has no default; it **MUST** be filled in.

Scanner Mode

Specifies the mode used to select channels, either Sequential (from start channel to stop channel) or Random from an external channel memory. The default is Sequential.

Start Channel

Appear only when the Scanner Mode is Sequential.

Stop Channel

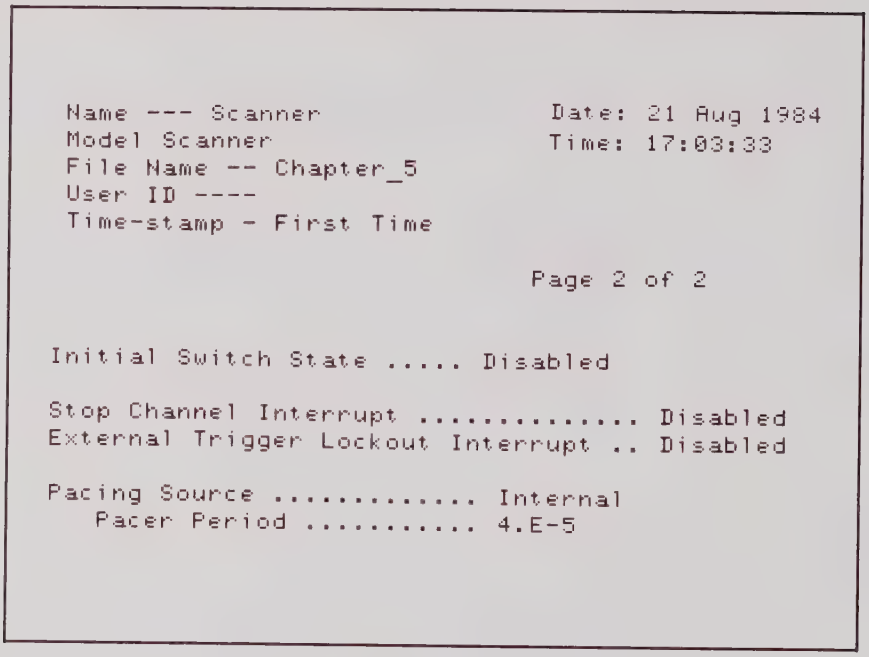
The Start channel specifies the first channel in the range of channels to be scanned. The legal range is from 0 to 1023. The default is 0. The Stop channel specifies the last channel in the range of channels to be scanned. The Scan Control card can be programmed to interrupt or indicate completion when the current channel is equal to the stop channel. The legal range is from 0 to 1023. The default is 15.

Memory Frame
Memory Slot
Memory Type

Appear only when the Scanner Mode is Random. This specifies external channel memory and therefore requires the Memory card's frame, slot and type. The Memory card/Memory Card System must be at the same select code as the 69750A Scan Control card. The Load__channels statement is available when random scanning with external channel memory. This statement will allow the external channel memory to be loaded from an array of channels (see Load__channels statement). Memory Frame specifies the Multiprogrammer frame which contains the 69790B or 69791A. The default frame is 0. Memory Slot indicates the location of the 69790B or 69791A within the specified frame. This parameter has no default. It **MUST** be filled in. Memory Type indicates whether a 69790B Memory card or 69791A Memory Card System is being used. The default memory type is 69790B.

Timeout

Specifies the time in seconds that will be allowed for the 69750A card to complete operations. The 69750A completes operation after it starts the on-board pacer. If the card does not complete within the time specified for Timeout, an error message (code 803) is generated. The legal range for Timeout is from 0.001 to 128,000 seconds. The default value is 5 seconds.



Menu Page 2 of 2

Initial Switch State

Indicates whether the Scanner card switches will be Enabled or Disabled after execution of the Init statement. The default, Disabled, holds all the switches open until execution of the Set_switches(Name,1) statement.

Stop Channel Interrupt

Specifies whether or not accessing the Stop Channel will interrupt the computer. Selecting "Enabled" allows an interrupt to be generated when the current channel address is equal to the Stop channel address. The Enable_intr statement must have been previously executed (see Enable_intr statement). The default is Disabled.

External Trigger Lockout Interrupt

Specifies whether or not the computer will be interrupted when the External Trigger Lockout condition occurs. Selecting Enabled allows an interrupt to be generated when the External Trigger Lockout is enabled via the Lockout input signal on the card edge or the execution of the Enable_lockout statement. The Enable_intr statement must have been previously executed (see Enable_intr statement). The default is Disabled.

Pacing Source

The pacing source can be provided by an external signal on the card edge, or by an internal pacer on the Scan Control card. The default is Internal. Selecting Internal brings up the next field, Pacer Period.

Pacer Period

Determines the scan rate for the internally paced scan. The period is specified in seconds from .000001 to 65,535. The default is .00004 seconds.

Programming Fundamentals

Only three statements are needed to program the 69750A Scan Control card via the Scanner menu. Each statement is described below.

Init(Name)

Sets up the initial conditions specified in the Scanner menu (Scanner Mode, Start Channel and Stop Channel or Memory Frame, Memory Slot and Memory Type, Pacing Source, Initial Pacer Period, Initial Switch State, Initial Interrupt Mask, and Timeout) and sets the Switch Mode default to Singlepoint. Interrupts from the card are disabled as if a Disable_intr statement were executed. For a scanning A/D application, Name is the name of the A/D card. For a buffered scanning A/D application, Name is the name of the Buffer menu. Note that this statement does not start the on-board pacer.

Load_channels(Name, Channels (•)[,Range])

Can be used only if the Scan Mode is Random with the external channel memory. This statement will allow the external channel memory to be loaded from the Channels(•) array which contains the previously entered channels to be scanned. The optional Range array parameter can be used only if the Scanner menu has been referenced from the 69759A A/D. This will make it easier to supply range information to the A/D card while providing channel information to the scanner. The Range array **MUST** be the same size as the Channel array. The data array(s) **MUST** be previously DIMensioned or ALLOCATEd in the user's program or a BASIC system error (#13) will be generated.

Start(Name)

This statement starts the on-board pacer. The Stop statement (see advanced programming statements) will stop the on-board pacer.

Random Scannng A/D Example

The previous 69759A Scanning A/D Advanced Programming example can be modified so that 10 channels will be randomly scanned every minute for 10 minutes. In order to do this, a 69790B Memory card has been added for external address memory. Figure C-3 in Appendix C shows the card inteconnections for this example (with the exception of $\overline{\text{GATE}}$ (k) of the 69759A connected to EXT (m) of the 69750A as shown in Figure 5-1). The following changes must be made to the menus in the previous example:

For the Scanner menu:
(named, Rndm_scan)

```
Scanner Mode.....Random
  Memory Frame.....0
  Memory Slot.....8
  Memory Type....69790B
```

For the 69759A A/D Converter card menu:
(named, Adc)

```
Scanner .....Rndm_scan
```

Random Scanning A/D Program Listing

```
10    COM /Names/ Adc,Rndm_scan
20    DIM Readings(0:99),Channels(1:10)
30    Configure("Menu_on","Rdm_config")
40    !
50    ! Load the channels array with the random
60    ! channels to be scanned.
70    !
80    DATA 10,1,6,50,12,15,35,41,9,25
90    READ Channels(*)
100   Minutes=0
110   ON CYCLE 60 GOSUB Take_a_reading
120 Continue:IF Minutes<>10 THEN Continue
130           ! Otherwise Stop
140   OFF CYCLE
150           ! Print total readings after 10 Minutes
160   PRINT Readings(*)
170   GOTO Done
180   !
190   ! Initialize the Memory card, A/D Converter,
200   ! and the Scan Control card each time readings are
210   ! taken.
220   !
230 Take_a_reading:Init(Adc)
240   !
250   ! Load the external channel address memory
260   ! with the channels.
270   !
280   Load_channels(Rndm_scan,Channels(*))
290   Index=10*Minutes
300   !
310    ! Initialize a Gate signal from the A/D card
320    ! which triggers the Scan Control card and begins the
330    ! internal pacer. Scan 10 specified channels and store
340    ! the resultant readings in the array.
350    !
360   Input_rblock(Adc,Readings(*),10,Index)
370   Stop(Rndm_scan)
380   Minutes=Minutes+1
390   RETURN
400 Done:END
```

Advanced Programming

The statements that allow special purpose programming of the Scan Control card via the Scanner menu are the same(excluding Init and Start) as those for the Scan Controller menu. To use the Scanner menu for more complex applications, first familiarize yourself with the card as described in the 69750A Operating Manual (HP Part No. 69750-90001), and then proceed to the Advanced Programming Section which follows the Scan Controller menu.

3478A Digital Multimeter

The 3478A Digital Multimeter is an HP-IB device. This requires a dedicated 98624A HP-IB Interface card in your Computer.

The 3478A Digital Multimeter provides measurement capability of DC or AC voltage, DC or AC current, or resistance (2 wire and 4 wire ohms). The 3478A employs an analog-to-digital conversion technique that integrates the input signal for a period of time (1/600, 1/60, or 1/6 of a second) to average out the effects of noise. Refer to the 3478A Operation's Manual (HP Part No. 03478-90003) for complete specifications and details about the hardware and the various modes of operation.

Menu Entries for the 3478A Digital Multimeter

The configuration and initial conditions for the 3478A function are specified on a two page menu. You are required to fill in various parameters on the menu in order to program the function.

```
Name --- Dig_multi                Date: 21 Aug 1984
Model 3478A Multimeter            Time: 17:15:14
File Name -- Chapter_5
User ID ----
Time-stamp - First Time

                                  Page 1 of 2


HP-IB Select Code ... 8
HP-IB Address ..... 22

Function ........... DC Volts
Range .............. Autorange
Trigger ............ Internal
Number of Digits .... 5.5
Autozero ........... On

Timeout ............ 5
```

Menu Page 1 of 2

HP-IB Select Code — Identifies the computer select code of the 98624A HP-IB interface. The legal range is from 8 to 31. The default value is 8.

HP-IB Address — Identifies the HP-IB address of the 3478A Digital Multimeter. The legal range is from 0 to 30. The default is 22.

Function — Specifies the 3478A function: DC Volts, AC Volts, DC Current, AC Current, 2 Wire Ohms, 4 Wire Ohms, or Extended Ohms. The default is DC Volts.

Range — Specifies the range of the 3478A: Autorange, .030, .300, 3, 30, 300, 3 k, 30 k, 300 k, 3 M, or 30 M. The ranges are specified without units (volts, amps, ohms) because the units are dependent on the function. The default value is autorange.

NOTE

Not all ranges are valid for a particular function. An invalid range for a given function defaults to the most sensitive range if the range specified is too low, or to the least sensitive range if the range specified is too high. No error will be generated.

Menu Page 1 of 2 (continued)

Trigger

Specifies the trigger mode of the 3478A (Internal or External). The default is Internal.

While in Internal trigger mode, the 3478A will take readings continuously. The Get statement will return the most recent available reading.

While in the External trigger mode, the 3478A will start a conversion on the falling edge of the external trigger signal. The Get statement will access the most recent available reading. If no reading is available, program execution will pause, waiting for the reading to become available. If a reading does not become available within the fixed time limit (5 seconds) assigned to a single HP-IB transaction, an HP-IB timeout error message (code 822) is generated. **Note** that HP-IB timeout is not the timeout that is specified in the configuration menu. Calling the Start or Input statements will cause any measurement currently in process to be aborted, and start a new conversion. Upon storing the resultant reading, the previous trigger mode is resumed.

Number of Digits

Specifies the number of digits displayed on the front panel of the 3478A (3.5,4.5, or 5.5). The default is 5.5.

This parameter affects the time required to perform a conversion cycle. A 3½ digit reading is produced by integrating the input signal for 1/600 of a second, and then converting it to a number. This is the fastest type of conversion. A 4½ digit reading is integrated for ten times as long (1/60 of a second) yielding rejection of 60 Hz power line noise. A 5½ digit reading is actually the average of ten 4½ digit readings. As can be seen, there is a trade off to be made between accuracy and reading rate.

Autozero

Specifies whether the Autozero mode of the 3478A is On or Off. The default is On. The 3478A will do an internal measurement cycle before every reading to insure that offsets due to changes in the temperature of the 3478A can be calculated out of the reading. By keeping the 3478A at a stable temperature and turning the Autozero off, the reading rate can be almost doubled without sacrificing accuracy.

Timeout

Specifies the time in seconds that will be allowed for the 3478A to complete an A/D conversion. If a Wait_for statement does not complete within this time period, Error Code 803 will be generated. The legal range is from 0.001 to 128,000 seconds. The default value is 5 seconds. The time required for the 3478A to complete a conversion depends upon the Number of Digits parameter selected. (**Note** that Timeout does not apply to Input, Input_fast, or Get statements; these are governed by a fixed HP-IB timeout value).

```
Name --- Dig_multi                Date: 21 Aug 1984
Model 3478A Multimeter            Time: 17:15:21
File Name -- Chapter_5
User ID ----
Time-stamp - First Time

                                  Page 2 of 2

Data Conversion ..... Standard



Data Ready Interrupt ..... Disabled
Front Panel SRQ Interrupt .. Disabled
```

Menu Page 2 of 2

Data Conversion

Specifies how the data values that are returned from the 3478A function in an Input, Input_fast, or Get statement are to be scaled. One of two options can be selected: "Standard" or "User".

1. "Standard" conversion is the default for the 3478A function. The data values returned represent the standard units of the function. Standard units for the 3478A depend upon the Function parameter selected.

2. "User" conversion allows the data values to be specified in terms of the application rather than in terms of the function's standard units. As described below, four additional entries appear on the menu when User Conversion is selected.

User's
P1 P2

Standard
P1 P2

Appear on the menu only when User conversion is selected. These entries allow you to fill in the ratio of User and Standard values so that the 14752A can convert the data returned from the multimeter into user defined units. The P1 default values are 0; the P2 default values are 1.

Menu Page 2 of 2 (continued)

Example: Assume that the 3478A is connected to a sensor that converts a pressure measured in pounds/sq. in. (psi) into a voltage: 0 psi is equal to 0 volts and 100 psi is equal to 10 millivolts. First, the 3478A's "DC Volts" Function is selected on the menu. In order to program in "psi" instead of in "volts", select the "User" Data Conversion Option and enter the following in the P1 and P2 parameters.

	P1 Entry	**P2 Entry**
User's	0 psi	100 psi
Standard	0 volts	.010 volts

Data Ready Interrupt Specifies whether or not the 3478A will interrupt the computer when a data ready condition occurs. The interrupt can be enabled or disabled; the default is disabled.

NOTE

The Data Ready and Front Panel SRQ interrupts are not actually enabled until the execution of the Enable_intr statement; see the "Advanced Programming" discussion.

Front Panel SRQ Interrupt Specifies whether or not the 3478A will interrupt the computer when the front panel SRQ button is pressed. This interrupt can be enabled or disabled; the default is disabled.

Programming Fundamentals

Only two 14752A statements are needed to program the 3478A Digital Multimeter. These statements are described below, followed by a simple example program.

Statement	Description
Init(Name)	Sets up the initial configuration specified in the menu (Function, Range, Trigger, Timeout, Number of Digits, and Autozero). Disables interrupts. Useful when changing configurations without triggering the 3478A to prepare for an external trigger.
Input(Name,Value)	Sets up the 3478A configuration according to the parameter specified in the menu (Function, Range, Trigger, Number of Digits, Timeout, and Autozero). Starts a conversion cycle and returns the resultant reading into the variable, "Value."

In the following program example, two functions on the 3478A will be used in the same program. The characteristics of a signal will be measured in terms of DC and AC magnitude. The signal to be measured is connected between the HI and LO Input (common) terminals on the 3478A. During the configuration phase, the DC Volts and AC Volts Functions on the same 3478A are set up on different Names as follows:

Name	Level
Function	DC Volts
Range	30 (volts are implied by the function)
Trigger	Internal
Number of Digits	5.5
Autozero	On
Name	Ripple
Function	AC Volts
Range	Autorange
Trigger	Internal
Number of Digits	3.5
Autozero	On

In this example, sixteen DC (Level) measurements and sixteen AC (Ripple) measurements are made and the results are stored in a 16 x 2 array:

```
10 COM /Names/ Level,Ripple
20 DIM Results(1:16,1:2)
30 Configure("Menu_on","ACDC")
40 System_init
50 FOR M = 1 TO 16
60 Input(Level,Results(M,1))
70 Input(Ripple,Results(M,2))
80 NEXT M
90 END
```

Advanced Programming

There are 14752A statements that allow special purpose programming of the 3478A. They are considered advanced because they require a deeper understanding of the interaction of the 14752A CAT Programming Package with the 3478A. These statements are useful for external triggering and interrupts, control of the 3478A display, and modification of the function, range, trigger mode, number of digits, and autorange mode of the 3478A during program execution.

Statement	Description
Start(Name)	Sets up the present configuration (Range, Function, etc.) and starts an A/D conversion.
Get(Name,Value)	Sets up the present configuration (Range, Function, etc.), reads the 3478A, and returns the reading in the variable, "Value." Used in conjunction with the Start statement or with the External Trigger signal. Note: the 3478A does not employ dual rank storage of data as do the Multiprogrammer input cards. A Get statement, for the 3478A, will pause until data is ready or the HP-IB t'meout occurs. A Get statement should be executed only when you know that data is ready to be read back; use the Wait_for or Check_done statements to determine when data is ready.
Set_data_ien(Name,Flag)	This statement programs the bit in the 3478A Interrupt Mask corresponding to the interrupt generated when a reading is available to be read by a Get statement. The mask does not take effect until the execution of the Enable_intr statement. A flag of 1 allows the condition to cause an interrupt; a flag of 0 prohibits it from doing so.
Set_fp_srq_ien(Name, Flag)	This statement programs the bit in the Interrupt Mask corresponding to the interrupt generated by pressing the front panel service request button. The mask does not take effect until the execution of the Enable_intr statement. A flag of 1 allows the front panel button to cause an interrupt; a flag of 0 prohibits it from doing so.

NOTE

While Front Panel SRQ is enabled to interrupt, you should not execute either a Check_done or a Wait_for statement on the 3478A since the Serial Poll operation may cause the interrupt service routine parameter "Reason" to be set incorrectly. As an alternate approach to handling data from the DMM, either use an Input statement (for internal trigger), or enable the 3478A to interrupt on Data Ready (for external trigger).

Advanced Programming (continued)

Statement	Description
Enable_intr(Name,Vector)	Allows the 3478A to signal the computer via the interrupt system at the end of its conversion cycle or by the front panel SRQ button. See Set_data_ien and Set_fp_srq_ien for details on setting the interrupt mask. See Chapter 10 for details on the use of the BASIC interrupt statements. **Note:** the Init and System_init statements will cause the interrupts to be disabled as if a Disable_intr statement was executed. If more than one Name is defined for the same 3478A, the Enable_intr statement for one Name has the effect of a disable_intr statement on the other Names. The parameter, Vector, identifies the function that caused the interrupt.

The 3478A can interrupt for any of several reasons. The parameter "Reason" in the "User_handler" service routine can be used to determine the cause of the interrupt, and thus direct the software to perform the proper operation. The values provided by "Reason" for each cause are binary-weighted, since more than one can occur at the same time. The two interrupts provide the following values in reason when they occur:

1-Data Ready
2-Front Panel SRQ

Disable_intr(Name)	Inhibits the 3478A from interrupting the computer. Meaningful only after the execution of an Enable_intr statement.
Check_done(Name,Flag)	Inquires about the state of the data ready signal of the 3478A. Returns a "1" in the variable, Flag, if data is ready; otherwise Flag is "0".
Wait_for(Name)	Suspends program execution until data is ready or a timeout occurs. Used to wait for the 3478A to complete its operations when triggered with the Start statement or an external trigger.
Get_conditions(Name, Value)	Serial Polls the 3478A and returns the resultant byte in the variable, "Value". The number returned depends on the binary-weighted conditions listed below:

Binary Weighted Value	Condition
1	Data Ready
2	"Name" not currently active
4	Syntax Error
8	Hardware Error
16	Front Panel SRQ
32	Cal Failed
64	SRQ
128	Power-on RESET

Advanced Programming (continued)

Example: If Get__conditions returns a 65, this indicates that data is ready and the 3478A is requesting service. Note: serial polling the 3478A will cause an SRQ to be cleared. If you have enabled interrupts on a 3478A and are expecting an interrupt, you should not execute any Get__condition statements.

Statement	Description
Start__fast(Name)	Sets up the configuration (function, range, etc.) and starts an A/D conversion without the usual settling time allowed for the input filters. Used only on AC functions and highest resistance ranges.
Input__fast(Name,Value)	Performs a Start__fast and returns the resultant reading into the variable, "Value".
Set__timeout(Name,Value)	Changes the value of timeout from the time specified in the menu to Value. The legal range for Value is from 0.001 to 128,000 seconds.
Set__disp__mess(Name, Message$)	Sends a message to the 3478A display of 12 or more characters. Also turns off display update which improves the reading rate. Message$ is a string. This can be useful for operator prompts.
Set__display__on(Name)	Returns the display of the 3478A to its normal mode.

Do not become alarmed if the 3478A's front panel annunciators do not change when any of the following "Set..." statements are programmed. Under certain conditions these statements will execute immediately. However, there are conditions which will cause the statements to be "remembered" by the 14752A but not executed until a later time. They **will** definitely be executed just before any of the following statements that use the same Name as the "Set..." statement: Init, Input, Input__fast, Get, Start, Start__fast, Enable__intr, and Disable__intr.

Statement	Description
Set__func__dcv(Name)	Causes the 3478A to read DC volts.
Set__func__acv(Name)	Causes the 3478A to read true rms AC volts.
Set__func__dca(Name)	Causes the 3478A to read DC Current.
Set__func__aca(Name)	Causes the 3478A to read true rms AC current.
Set__func__2wo(Name)	Causes the 3478A to read ohms as measured by the 2 wire ohms method.
Set__func__4wo(Name)	Causes the 3478A to read ohms as measured by the 4 wire ohms method.
Set__func__exo(Name)	Causes the 3478A to read ohms by the 2 wire ohms method with a 10 M ohm shunt.

Advanced Programming (continued)

Statement	Description
Set_range(Name, Range)	Sets the range of the 3478 to the value specified by the "Range" parameter. The range **MUST** be one of the following: .030, .300, 3, 30, 300, 3 k, 30 k, 300 k, 3 M, 30 M, or an error (#847) will be generated.
Set_rng_auto(Name)	Allows the 3478A to select the proper range to make the measurement.
Set_trigger_int(Name)	Causes the 3478A to initiate a new conversion upon completion of the last. Therefore, a recent reading is always available.
Set_trigger_ext(Name)	Causes the 3478A to initiate a new conversion upon receipt of an external trigger pulse.
Set_digits_3(Name)	Causes the 3478A to take 3½ digit readings (fastest).
Set_digits_4(Name)	Causes the 3478A to take 4½ digit readings (best compromise).
Set_digits_5(Name)	Causes the 3478A to take 5½ digit readings (best noise rejection).
Set_autozero(Name,Flag)	If Flag = 1, causes the 3478A to perform an internal calibration cycle before each reading (continuous thermal compensation). If Flag = 0, eliminates the internal calibration cycles. If you keep the 3478A at a constant temperature, you can double the reading rate.

Chapter 6
Analog Output Functions

Introduction

An Analog Output Function provides a continuous output which can be programmed within a specified range. The type of output generated by an Analog Output function can take various forms. The basic electrical quantities of resistance, voltage, and current are available. In addition to the small signal or low power voltage and current outputs, higher power voltage and current can be attained through the programming of various power supplies.

The Multiprogrammer provides a number of Analog Output Functions. The 69700A-69706A Resistance Output cards provide a programmable resistance output value that can be used in various applications including the programming of a number of HP power supplies which have been equipped with Remote Programming Option 040. The 69709A Power Supply Programming card can be used to program either the 6011A, 6012A, 6012B, 6023A, or 6024A Power Supplies equipped with Option 002. Two Digital-to-Analog Cards, the 69720A and 69721A, provide a programmable voltage and a programmable current output, respectively. The 69720A and 69721A cards can be combined with the 69790B Memory card or 69791A Memory Card System, to form a Buffered Digital to Analog (D/A) multiple card function. This function can be configured via the buffer menu as described in Chapter 9. The desired function is associated with the user's specified name using the Name selection and Function Selection menus as described in Chapter 3. The following paragraphs describe the menus and 14752A statements that are used to program the 69700A-69706A, 69709A, and 69720A/69721A Analog Output functions.

69700A-69706A Resistance Output

These cards provide a programmable value of resistance as their output. They are often used to program power supplies such as HP power supplies equipped with Option 040, Remote Programming. Twelve magnetically shielded, mercury wetted, reed relays select the resistance values by modifying the value of a series string of high-accuracy, binary weighted resistors. The 69700A card is supplied without any output resistors; customers may choose and install their own resistors. Refer to the 69700A-69706A Resistance Output Cards Operating Manual, HP Part No. 5950-1924, for the hardware details on each model.

Binary-weighting refers to the ratio between the resistor values which make up the resistor string. There is a X2 relationship between adjacent resistor values. For example, starting at the bottom (LSB) of a string with a value of 1, the next values are 2,4,8,16 and so on for the 12 resistors in the string. The following table lists the Resistance Output cards with the maximum resistance values and resolution (LSB) of each model.

Model	Maximum Resistance (ohms)	Resolution/LSB (ohms)
69700A	Unloaded	-
69701A	8,190	2
69702A	30,712.5	7.5
69704A	40,950	10
69705A	81,900	20
69706A	204,750	50

The Resistance Output cards are shipped with a ten ohm resistor connected in series with the output terminals. When a Resistance Output card is being used to control HP Power Supplies with Option 040 installed, a factory calibration is performed to insure that the power supply can be accurately set to zero voltage and to the maximum rated output voltage or current when programmed. The ten ohm resistor permits the user to maintain the zero calibration despite line losses between the card and the remotely located power supply. When a Resistance card is not controlling a HP Power Supply, the series 10 ohm resistor can be bridged by installing card jumper W1; see the Resistance Cards Operating manual for details.

Menu Entries for the Resistance Output Cards

The configuration and initial conditions for the 69700A-69706A Resistance Output cards are specified on a two page menu. The menus for each card are identical except the 69700A card includes an additional parameter, LSB Resistance, on menu page 1. The user is required to fill in the various parameters on the menu in order to program the function. Instructions on how to use the menus and additional information on most of the parameters are given in Chapter 3. The following discussion uses the 69700A Resistance Output card's menu as an example in describing the various Resistance function parameters.

```
Name --- Resistance                Date: 21 Aug 1984
Model 69700A Resistance            Time: 17:18:07
File Name -- Chapter_6
User ID ----
Time-stamp - First Time

                                Page 1 of 2

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot ..........

LSB Resistance .. 1


Timeout ......... 5
```

Menu Page 1 of 2

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ..... 8
HP-IB Address ......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Menu Page 1 of 2 (continued)

Frame

Specifies the Multiprogrammer frame which contains the Resistance Output card. The legal range is from 0 to 7. The default value is 0.

Slot

Identifies the location of the Resistance Output card within the frame. The legal range is from 0 to 15. This parameter **MUST** be filled in; it has no default value.

LSB Resistance

This parameter applies only to the 69700A Resistance Output card. It specifies the minimum value (LSB) of resistance in ohms of the customer installed resistors. The default value is 1.

Timeout

Specifies the time in seconds that will be allowed for the Resistance Output card to complete its operation; that is, change its output resistance. If an Output or Wait_for statement does not complete within this time, a Timeout error message (error code 803) will be generated. the legal range for Timeout is from 0.001 to 128,000 seconds. The default value is 5 seconds. The Resistance Output card completes operations in 6 milliseconds.

```
Name --- Resistance                Date: 21 Aug 1984
Model 69700A Resistance            Time: 17:18:33
File Name -- Chapter_6
User ID ----
Time-stamp - First Time

                                   Page 2 of 2


Data Conversion ..... Standard


Run Time Limit Checking ... No


Initial Value ..... 0
```

Menu Page 2 of 2

Data Conversion

Specifies how the data value sent to the Resistance Output card by an Output or Preset statement is to be scaled. One of three options can be selected: Standard, User, or None.

1. "Standard" conversion is the default for a Resistance Output card. The data values sent to the card represent the standard units (ohms) of the card. For example, a value of 15360 sent to the 69705A card would result in an output resistance of 15,360 ohms.
2. "User" allows the card data to be specified in terms of the application rather than in terms of the card's standard units (ohms). This means that you can program in user defined units and have the data scaled when it is sent to the card. As shown below, four additional entries appear on the menu when User conversion is selected.

Menu Page 2 of 2 (continued)

User's
P1 P2

Standard
P1 P2

Appear on the menu only when User conversion is selected. These entries allow you to fill in the ratio of User and Standard values so that the 14752A can convert the user defined data into values which are recognized by the card. The P1 default values are 0; the P2 default values are 1.

Example: Assume that the 69701A card is used to program the output of a 6259B Option 040 Power Supply. The 69701A has an LSB (resolution) of 2 ohms while the 6259B has a resolution of 10 mV. The 6259B has a maximum output voltage of 10.0 volts which requires the 69701A to be set no higher than 2000 ohms. In order to program the 69701A directly in volts instead of in ohms, select User Data Conversion and enter the following in the P1 and P2 parameters.

User's	P1.	.010	P2.	10.0
Standard	P1.	2	P2.	2000

3. "None" conversion specifies that the data which is sent to the Resistance Output card represents the number of LSB's. For the 12-bit Resistance Output card, the range of this data is from 0 to 4095. For example, sending the value 512 with "None" conversion selected, programs the output of the 69701A card to 1024 ohms; 512 X LSB; or 512 X 2 ohms. This is the fastest way of sending data to the Resistance Output card. An Output statement will execute 20% to 30% faster with the "None" conversion than with 'Standard" conversion; however, the "Standard" or "User" conversion provides a much easier way to send data to the Resistance Output function.

```
Name --- Resistance                Date: 21 Aug 1984
Model 69700A Resistance            Time: 17:19:35
File Name -- Chapter_6
User ID ----
Time-stamp - First Time

                                 Page 2 of 2


Data Conversion ..... Standard


Run Time Limit Checking ... Yes
        High Limit .........
        Low Limit .........

Initial Value ..... 0
```

Menu Page 2 of 2

Menu Page 2 of 2 (continued)

Run Time Limit Checking

If Yes is selected, the value sent to the Resistance Output card will be checked against the High and Low Limits that you specify. The default for this parameter is off (No).

High Limit

Appears only when Run Time Limit Checking is on (Yes). Specifies the upper bound for the value programmed. The value entered must be greater than the Low Limit entered. Error code 802 will be generated when a programmed value is greater than the High Limit entered. This value **MUST** be entered in the same units as selected under Data Conversion. This parameter **MUST** be filled in; it has no default value.

Low Limit

Appears only when Run Time Limit Checking is on (Yes). Specifies the lower bound for the value programmed. The value entered must be lower than the High Limit entered. Error code 802 will be generated when a programmed value is lower that the Low Limit entered. This value **MUST** be entered in the same units as selected under Data Conversion. This parameter **MUST** be filled in; it has no default value.

Initial Value

Specifies the value which will be sent to the Resistance Output card when it is initialized by an Init or System_init statement. This value **MUST** be entered in the same units as selected under Data Conversion. The default value is 0 standard Units (0 ohms).

Programming Fundamentals

Only two 14752A statements are required to program the Resistance Output function. Each statement is described below and then a simple example program is given. For details concerning the syntax requirements for these statements, refer to Chapter 3.

Statement	Description
Init(Name)	Sets the output resistance on the named Resistance Output card to the Initial Value specified in the menu. Init also clears the EOP (End-of-Process) signal at the card's edge connector, disables card interrupts, and resets Timeout to the values specified in the menu.
Output(Name,Value)	Sets the output resistance on the named Resistance Output card to the value specified in the variable "Value".

Amplifier Gain Programming Example

A 69700A-69706A Resistance Output card can be used in applications which require a programmable, high-precision resistance output. For example, suppose that a certain amplifier has a programmable gain feature. The gain can be varied in proportion to the variation in resistance. A resistance value of 100 ohms produces a gain of 1 and a value of 10,000 ohms produces a gain of 100. Assume that the 69704A card (0 to 40950 ohms, 10 ohms resolution) supplies the programmable resistance. In order to program the 69704A directly in terms of the amplifier gain instead of in ohms, select User Data Conversion on the menu and make the following entries in P1 and P2.

	P1 Entry	P2 Entry
User's	Gain = 1	Gain = 100
Standard	100 ohms	10,000 ohms

By using the "User" conversion, the data sent to the Resistance Output card is in terms of the amplifier gain i.e., a value of 12.5 represents an amplifier gain of 12.5. The 14752A will convert the data value 12.5 into the standard data units required by the 69704A Resistance Output card.

In the following programming example, the 69704A Resistance Output card is identified with the name, "Amplifier_gain". During the configuration phase, the name is defined, the initial output is set, and the user conversion factor is entered as indicated above.

```
10  COM/Names/Amplifier_gain
20  Configure("Ask_me","Myfile")
30  !
40  ! Initialize the card
50  !
60  Init(Amplifier_gain)
70  !
80  ! Change the amplifier's gain to
90  ! a value of 10.5
100 !
```

```
110 Output(Amplifier_gain,10.5)
120 !
130 ! Change the amplifier's gain again but use
140 ! a variable called "New_Value"
150 !
160 New_value=47.5
170 !
180 Output(Amplifier_gain,New_value)
190 END
```

Power Supply Programming Example

A 69700A-69706A Resistance Output card can be used to resistance program the output voltage of a variety of power supplies such as the HP family of power supplies which have been equipped with Option 040, Remote programming. A selection guide for the HP family of programmable power supplies and the Multiprogrammer Resistance Output cards is included in the Multiprogrammer Series II I/O Cards Data Sheet, HP Part No. 5952-4090.

Many electronic systems require that their multiple power supplies be turned on in a particular order. The following program uses two Resistance Output cards named "Vcc" and "V_plus" to sequentially program two power supplies. The "Vcc" supply provides a low positive voltage and the "V_plus" provides a higher positive voltage. The low voltage supply is turned on first, followed by V_plus. Both of the Resistance Output cards are configured for User conversion so that they can be programmed in volts instead of in ohms; see the User Conversion example given on the description of Menu Page 2 of 2.

```
10  COM/Names/Vcc,V_plus
20   Configure("Ask_me","MYFILE")
30  !
40  ! First, initialize the two power
50  ! supplies to their initial safe state(0 Volts)
60  !
70  System_init
80  !
90  ! Set the "Vcc" supply to 5.00 volts
100 !
110 Output(Vcc,5.00)
120 !
130 ! Set the "V_plus" supply to +15.0 volts
140 !
150 Output(V_plus, 15.00)
160 !
170 END
```

Advanced Programming

There are 14752A statements which allow special purpose programming of the Resistance Output cards. They are considered to be advanced programming statements because you require a good understanding of the Multiprogrammer System to put these statements together to form a meaningful program. In general, these statements are used only when precise control of timing is a concern. They allow features like parallel output of more than one function, interrupts, and external triggering (see Chapter 10 for details). The advanced programming statements that can be used to program a Resistance Output card are described below.

Statement	Description
Preset(Name,Value)	Sends a resistance value to the named card but does not cause the output to change. The Preset statement is used as a setup operation for the Start statement or for an external trigger.
Start(Name1...Name8)	Sets the output on the named card(s) to the resistance specified by the last value sent to the card. All named (up to 8) functions are "started" at the same time.
Wait_for(Name)	Suspends program execution until the Resistance Output has completed operations or until a timeout has occurred. Used if you want to wait for the card to finish when it has been started with an External Trigger signal or a Start statement. The Timeout parameter in the menu specifies the timeout period. Error code 803 is generated if the timeout period elapses before the card completes operations.
Check_done(Name,Flag)	Inquires about the state of the Resistance Output card's operation. If the card has completed, the value of Flag returned is a "1". If the card has not completed, the value returned is a "0". Once a value of "1" is returned, Check_done will subsequently return a "0" until the function is started again and completes another operation.
Set_timeout(Name,Value)	Changes the value of Timeout from the time specified in the menu to Value. Time is specified in seconds, from 0.001 to 128,000.
Enable_intr(Name,Vector)	Allows the Resistance Output card to signal the computer via the interrupt system when it has finished operations. This statement must be used in conjunction with the interrupt statements as described in Chapter 10. The "Vector" variable identifies the function that caused the interrupt. Interrupts are initially disabled by the Init or System_init statements.
Disable_intr(Name)	Inhibits the Resistance Output card from signaling the computer when it has completed operations. It is meaningful only when an Enable_intr statement has already been sent. As noted above, the Init and System_init statements will also cancel a previously issued Enable_intr statement.

69709A Power Supply Control

The 69709A Power Supply Control card provides all of the features required to easily program, control, and monitor Model 6010A, 6011A, 6012A, 6012B, 6023A and 6024A Power supplies equipped with Option 002. The 69709A also maintains 250 V isolation between analog input signals and digital data common, thereby eliminating a potential source of group loop problems. Refer to the 69709A Operating Manual, HP Part No. 69709-90001, for complete hardware details.

CAUTION

Unless you are using a HP 6012A or 6024A, you must make some jumper changes on your 69709A Card. These changes consist of removing jumpers W1 and W4 and installing jumpers W2 and W3. Refer to Appendix A of your 69709A manual for further details. This change must be done for the power supply to wake up with remote control enabled.

The output voltage and output current values can be programmed with 0.1% resolution allowing precise control over the power supplies output range. The power supply's output voltage and output current can be read with 10-bit resolution permitting evaluation of the actual load conditions.

Menu Entries for the 69709A

The configuration and initial conditions for the 69709A Power Supply Control card are specified on a four page menu. You must specify various parameters on the menu before you can program the card. Instructions on how to use the menus and additional information on most of the parameters are given in Chapter 3. The 69709A Power Supply Control function's parameters are described as follows:

```
Name --- Pwr_sup                    Date: 21 Aug 1984
Model 69709A P/S Control            Time: 17:20:02
File Name -- Chapter_6
User ID ----
Time-stamp - First Time

                                  Page 1 of 4
Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ............... 0
Slot ................
Full-scale Voltage ... 60
Full-scale Current ... 10
Initial Voltage ..... 0
Initial Current ..... 1
Timeout ............. 5
```

Menu Page 1 of 4

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer type ... HP6942A
HP-IB Select Code ...... 8
HP-IB Address .......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the 69709A Power Supply Control card. The legal range is from 0 to 7. The default is 0.

Menu Page 1 of 4 (continued)

Slot

Identifies the location of the Power Supply Control card within the frame. The legal range is from 0 to 15. This parameter **MUST** be filled in; it has no default value.

Full-scale Voltage

Specifies the maximum full-scale output voltage (in volts) of the controlled power supply. For the HP Model 6012A, 6012B and 6024A Power Supplies, the full-scale voltage is 60 volts. For the HP Model 6023A and 6011A Power Supply, the full-scale voltage is 20 volts. The default is 60 volts.

Full-scale Current

Specifies the maximum full-scale output current (in amps) of the controlled power supply. The full-scale current for the 6012A and 6012B is 50 A while of the 6023A is 30 A. The 6024A is 10 A and the 6011A is 120 A. This parameter is used by the 14752A to identify the power supply model. The default is 10 amps.

Initial Voltage

Specifies the output voltage that will be programmed by the 69709A card when the function is initialized. **MUST** be entered in the same units as selected under Data Conversion. The default value is 0 volts.

Initial Current

Specifies the output current that will be programmed by the 69709A card. when the function is initialized. **MUST** be entered in the same units as selected under Data Conversion. The default is 1 amp. Note that the default current value of 1 amp provides some margin for the power supply to achieve a 0 volt output in a reasonable amount of time without tripping the Overvoltage Protection (OVP) circuit.

Timeout

Specifies the length of time (in seconds) that will be allowed for the 69709A card (or power supply) to complete its operation and signal the computer.

If the Output Sampling parameter (see Menu page 2 of 4) is not selected, Timeout specifies the length of time that will be allowed for the 69709A card to send the programmed value to the power supply. This operation usually takes about 500 microseconds(.0005 seconds). If the Output Sampling parameter is selected, Timeout specifies the length of time that will be allowed for the power supply's output to reach 99.7% of the programmed value. The time required for the power supply output to reach its programmed value depends upon the particular power supply and the output load. If an Output or Wait_for statement does not complete its operation (defined above) within the required time, Error code 803 will be generated. The legal range for timeout is from 0.001 to 128,000 seconds. The default value is 5 seconds.

```
Name --- Pwr_sup                    Date: 21 Aug 1984
Model 69709A P/S Control            Time: 17:20:28
File Name -- Chapter_6
User ID ----
Time-stamp - First Time

                                  Page 2 of 4

Voltage Data Conversion .... Standard


Current Data Conversion .... Standard


Output Sampling ............ On
    Initially Sample .... Voltage
```

Menu Page 2 of 4

Voltage/Current Data Conversion

Specifies how the data value is to be scaled for the Output, Preset, Set_v_limit, Set_i_limit, and Input statements. One of three data conversion options can be selected: Standard, User, or None.

1. "Standard" Conversion is the default for the 69709A card. The data values which are sent to or returned from the card represent the standard units (volts or amps) of the card. For example, a data value of 25.6 represents an output voltage of 25.6 volts.

2. "User" conversion allows the data to be specified in terms of the application rather than in terms of the card's standard units (volts or amps). This means that you can program in user defined units and have the data scaled for you when it is sent to (e.g. Output statement) or returned from (Input statement) the card. User's P1 and P2 parameters as well as Standard P1 and P2 parameters appear on the menu when "User" conversion is selected.

Menu Page 2 of 4 (continued)

User's
P1 P2
Standard
P1 P2

Appear on the menu only when User conversion is selected. These values allow you to fill in the ratio of User and Standard values so that the 14752A can convert the user defined values into values that are recognized by the card.

Example: Assume that a DC motor is powered by a 6012A Power Supply. The motor requires 12.5 volts to run at a speed of 1200 RPM and there is a linear relationship between the voltage and RPM. The 69709A card could program the power supply using the "Standard" conversion option. The number 12.5 would then represent a motor speed of 1200 RPM. To change the number of volts to RPM, another calculation is required. However, by using the "User" Voltage conversion option, the 69709A card can be programmed in units of motor speed instead of in volts; e.g. a value 1200 would represent a motor speed of 1200 RPM. In order to do this, you would select "User" Data Conversion and enter the following in the P1 and P2 parameters.

	P1 entry	**P2 entry**
User's	0 RPM	1200 RPM
Standard	0 volts	12.5 volts

3. "None" conversion specifies that the data which is sent to (or returned from) the card represents a percentage of the full-scale output voltage (or current) of the power supply. The data has a range from 0 to 1023 which translates to 0% to 102.3% full-scale output. (There is a slight overrange). An Output or an Input statement will execute 20% to 30% faster with "None" conversion than with "Standard" conversion; however, the "Standard" or "User" conversion provide an easier way to program the card.

Output Sampling

Specifies whether program execution is to be suspended until the output of the power supply has reached 99.7% of the programmed output. Useful when the supply is providing power for loads that have a long time constant (e.g. capacitive loads). Output Sampling is enabled by specifying On which is the default value. The output voltage or output current can be selected for initial sampling which affect completion of just the Output_voltage and Output_current statements, respectively.

```
Name --- Pwr_sup                    Date: 21 Aug 1984
Model 69709A P/S Control            Time: 17:20:40
File Name -- Chapter_6
User ID ----
Time-stamp - First Time

                                 Page 3 of 4

Run Time Limit Checking ... No




Initial Voltage Programming Source .. Remote
Initial Current Programming Source .. Remote
Initial Output State ................ On
```

Menu Page 3 of 4

Run Time Limit Checking

If Yes is selected, the voltage output and current output values sent to the 69709A card by the Output, Preset, Set_i_limit, and Set_v_limit statements will be checked against the High and Low Limits that you set. The default value for this parameter is off(No).

Voltage High Limit

Used only when Run Time Limit Checking is on (Yes). It specifies the upper bound for programmed output voltage values. The value entered must be greater than the Voltage Low Limit entered. Error code 802 will be generated when a programmed value is greater than this limit. This value **MUST** be entered in the same units as selected under data conversion. This parameter **MUST** be filled in it has no default.

Menu Page 3 of 4 (continued)

Voltage Low Limit

Used only when Run Time Limit Checking is on (Yes). It specifies the lower bound for programmed output voltage values. The value entered must be lower than the Voltage High Limit entered. Error code 802 will be generated when a programmed value is less than this limit. This value **MUST** be entered in the same units as selected under data conversion. This parameter **MUST** be filled in; it has no default.

Current High Limit

Used only when Run Time Limit Checking is on (Yes). It specifies the upper bound for programmed output current values. The value entered must be greater than the Current Low Limit entered. Error code 802 will be generated when a programmed value is greater than this limit. This value **MUST** be entered in the same units as selected under Data Conversion. This parameter **MUST** be filled in; it has no default value.

Current Low Limit

Used only when Run Time Limit Checking is on (Yes). It specifies the lower bound for programmed output current values. The value entered must be less than the Current High Limit entered. Error code 802 will be generated when a programmed value is less that this limit. This value **MUST** be entered in the same units as selected under Data Conversion. This parameter **MUST** be filled in; it has no default value.

Initial Voltage Programming Source

Identifies where the initial output voltage programing information is to come from. Either Remote (from the 69709A card) or Local (Power Supply front panel controls) can be selected. The default is Remote.

Initial Current Programming Source

Identifies where the initial output current programing information is to come from. Either Remote (from the 69709A card) or Local (Power Supply front panel controls) can be selected. The default is Remote.

Initial Output State

Specifies whether the output of the power supply, controlled by the 69709A card, is to be enabled or disabled at the time the function is initialized. Either On (enabled) or Off (disabled) can be selected. The default is On.

```
Name --- Pwr_sup                    Date: 21 Aug 1984
Model 69709A P/S Control            Time: 14:04:14
File Name -- CHAPTER_6
User ID ----
Time-stamp - First Time

                                 Page 4 of 4

Constant Voltage Interrupt .... Disable
Constant Current Interrupt .... Disable
Unregulated Output Interrupt .. Disable
OVP Interrupt ................. Disable
Over Temperature Interrupt .... Disable
AC Fault Interrupt ............ Disable
Read Back Update Interrupt .... Disable
End Cycle Interrupt ........... Disable
```

Menu Page 4 of 4

The parameters on this page specify whether various operating conditions of the power supply that are detectable by the 69709A Power Supply Control card should interrupt the computer when they occur. Each interrupt can be enabled or disabled; the default is disabled. For example, if the power supply goes from the Constant Voltage mode into the Constant Current mode due to a change in the power supply's load, and the Constant Current Interrupt is enabled, the 69709A card can notify the controller. Interrupt handling is described in Chapter 10.

NOTE

Care should be exercised when using the interrupt system with the 69709A Power Supply Control function. Interrupts can be generated during the normal operation of the power supply which do not necessarily indicate that a fault has occurred. For example, if the power supply is programmed to a higher output level, the Unregulated Interrupt and the Constant Current Interrupt might be generated until the output has stabilized at the new output level. These can be considered "nuisance" interrupts. Your interrupt handling software routine should take them into account.

Menu Page 4 of 4 (continued)

Constant Voltage Interrupt

Notifies the computer that the power supply is operating in the Constant Voltage mode. You should enable this interrupt if, during normal operation, you expect the supply to be operating in the Constant Current mode and you wish to be notified if its mode changes to the Constant Voltage mode.

Constant Current Interrupt

Notifies the computer that the power supply is operating in the Constant Current mode. You should enable this interrupt if, during normal operation, you expect the supply to be operating in the Constant Voltage mode and you wish to be notified if its mode changes to the Constant Current mode.

Unregulated Output Interrupt

Notifies the computer that the power supply is in neither the Constant Voltage mode nor the Constant Current mode. This may be due to changes in the load connected to the supply.

OVP Interrupt

Notifies the computer that the Over Voltage Protection Circuit has been tripped and the supply outputs are shut down.

Over Temperature Interrupt

Notifies the computer that certain components within the power supply have exceeded a critical temperature and the supply outputs are shut down.

AC Fault Interrupt

Notifies the computer whenever the input ac power has been lost for any period of time. This is necessary since the power supply may not be connected to the same ac power main as the computer.

Read Back Update Interrupt

Signals the computer whenever the output voltage and the output current values sampled from the power supply have been updated. Typically this will occur every 100 milliseconds and can be used to closely monitor the supply outputs.

End Cycle Interrupt

Indicates that the 69709A card has completed a voltage and current programming operation; i.e., it has changed its control outputs to the power supply.

Programming Fundamentals

Only three 14752A statements are needed to program the 69709A Power Supply Control card. Each statement is described below and then a simple example is given. For details concerning the syntax requirements for these statements, refer to Chapter 3.

Statement	Description
Init(Name)	Loads the 69709A Power Supply Control card with the initial values and interrupt conditions specified in the menu, and programs the power supply outputs accordingly. Regardless of output sampling, the Init statement does not wait for the power supply outputs to achieve the programmed values.

If the power supply is being used as a voltage source (the usual manner), the current output will be programmed to an upper limit value which is not to be exceeded and the voltage programmed to the actual output desired. This is done by programming the following two statements:

Preset_current (Name,Value)	Formats the data in the variable "Value" according to the Data Conversion option specified in the menu and sends the data to the 69709A card's output current register. The statement completes immediately. The output of the power supply, however, is changed to this new value only when an Output_voltage or Start statement is executed, or an External Trigger signal is received.
Output_voltage (Name,Value)	Formats the data in the variable "Value" according to the Data Conversion option specified in the menu and sends the data to the 69709A card's output voltage register. Then the data in both the output current register and the output voltage register is sent to the power supply which will then begin to assume the new values. If Output sampling is enabled, the Output_ voltage statement will not complete until the power supply's output achieves the programmed value or Timeout occurs. If Output sampling is disabled, the Output_voltage statement completes immediately.

Power Supply Control Example

The features designed into the 69709A Power Supply Control card allow it to control the operation of 6011A, 6012A, 6012B, 6023A, or 6024A Option 002 Power Supplies. Cable assembly 69709-60001 may be used to connect the 69709A card to the Option 002 board in a 6012A, 6023A, or 6024A Power Supply. Hardware details concerning the 69709A card and the 6012A, 6023A and 6024A Power supplies are given in the applicable hardware manual.

The following example illustrates how to program the 69709A card to control a power supply. The 69709A card is identified with the name "Bias" in this example. During the configuration phase, the Name is identified and the initial conditions are set. These initial conditions include the initial output voltage and current. The "Standard" conversion option is selected so that the power supply can be programmed in terms of output voltage.

```
10  COM/Names/Bias
20  Configure("Ask_me","Myfile")
30  !
40  ! Initialize the power supply using the
50  ! initial conditions defined during the
60  ! configuration phase
70  !
80  Init(Bias)
90  !
100 ! Change the output of the "Bias"
110 ! Power Supply to 12.6 volts
120 !
130 Output_voltage(Bias,12.6)
140 END
```

Power Supply Sequencing Example

Many electronic systems require that their multiple voltage power supplies be turned on in a controlled sequence. For instance, power supply No. 2 should not be turned on until power supply No. 1 has settled to within a certain percentage of its regulated output. The 69709A card can be programmed by the 14752A to properly sequence multiple power supplies.

In this example, three power supplies controlled by three 69709A cards provide −5.00 volts, +5.00 volts, and +12.00 volts to a particular system. The +5.00 volt supply is identified by the Name "Vcc"; the −5.00 volt supply by the name "V_minus"; and the +12.00 volt supply by the name "V_plus". To insure that each power supply is properly turned on, the Output Sampling parameter is specified (Yes) in the menu for each named 69709A card during the configuration phase. The programming example is as follows:

```
10  COM/Names/Vcc,V_plus,V_minus
20  Configure("Ask_me","MY_FILE")
30  !
40  ! Initialize the three power supplies
50  ! to 0 volts output, 1 amp maximum current
60  !
70  System_init
80  !
90  ! Set output of Vcc supply to 5.00 volts with 4 amps
100 ! maximum current. Program will pause at each Output statement
110 ! until the supply has reached its final(regulated) value
120 !
130 Preset_current(Vcc,4.0)
140 Output_voltage(Vcc,5.00)
150 !
160 ! Now that Vcc is set, set the V_minus
170 ! supply to −5.00 volts and wait for it to settle
180 !
190 Output_voltage(V_minus, −5.00)
200 !
210 ! With the Vcc and V_minus supplies set to their
220 ! proper outputs, the V_plus supply is set to 12.00 volts
230 !
240 Output_voltage(V_plus,12.00)
250 !
260 ! All three supplies have reached their proper output
270 ! levels in the prescribed sequence.
280 !
290 END
```

Advanced Programming

There are 14752A statements which allow special purpose programming of the 69709A Power Supply Control card. They are considered advanced programming statements because you need a good understanding of the Multiprogrammer System in order to use them in a meaningful program.

In general, these statements are used only when precise control of the power supply's output is a concern. They allow features such as parallel operation of the power supply in conjunction with other functions, checking status conditions, and interrupts. All of these features are explained in Chapter 10. The advanced programming statements for the 69709A card are described as follows:

Statement	Description
Input(Name,Voltage__value, Current__value)	Reads the actual output voltage and current of the power supply. The data is formatted according to the Data Conversion option selected and the data values are returned in the two variables, Voltage__value and Current__value.
Preset__voltage (Name,Value)	Formats the data in the variable "Value" according to the Data Conversion option specified in the menu and sends the data to the 69709A card's output voltage register. The Preset__voltage completes immediately. The output of the power supply, however, is changed only when an Output__current or Start statement is executed or an External Trigger signal is received.
Output__Current (Name,Value)	Sends a Value of current to the Named function and programs the power supply with this current and the preset voltage. If output sampling is "on", current is the parameter sampled.
Start(Name 1,...,Name8)	Sets the output(s) on the controlled power supply(s) to the last value of output voltage or current sent to the card. Up to 8 functions can be included in this statement. All names included are "started" in parallel; that is, all actions will occur simultaneously. The Start statement does not wait for the 69709A to complete its operation, therefore it finishes quickly. With a Start statement, the program will continue without waiting for the 69709A card to complete its output operation.
Get__conditions (Name,Value)	Returns a value which represents the current state of all conditions of the power supply. The statement reads the 6 conditions that can also be obtained via the "FN" functions that are described later. Each condition is assigned a binary-weighted value as follows:

Condition	Value
Constant Voltage Mode	1
Constant Current Mode	2
Unregulated output	4
Over voltage	8
Over temperature	16
AC Fault	32

These values indicate that the corresponding condition is occurring in the power supply.

Advanced Programming (continued)

Statement	Description
Enable_intr (Name,Vector)	Allows the 69709A card to signal the computer via the interrupt system when it has completed operations, detected a fault condition, or a change in the power supply's output mode has occurred. The conditions which are enabled to cause an interrupt are as specified in the menu or as modified in any "Set_...ien" statement. This statement must be used in conjunction with the interrupt statements as described in Chapter 10. The "Vector" variable identifies the card that caused the interrupt. An Init, System_init, or Disable_intr statement will cancel a previously issued Enable_intr statement. The Power Supply Control function differs from other functions in that, once an interrupt has occurred and has been serviced, the Enable_intr statement must be re-executed in order for any future interrupts to occur. This allows you to modify the conditions which can cause interrupts prior to re-enabling them.
Disable_intr (Name)	Inhibits the 69709A card from signaling the computer when an interrupt condition occurs.
Wait_for(Name)	Suspends program execution until the 69709A card (or power supply) has completed operations or a timeout has occurred. Used if you want to wait for the card (or power supply) to finish when it was started by an External Trigger or Start statement. The Timeout parameter in the menu specifies the timeout period. Error code 803 is generated if the timeout period elapses before the card completes operations.
Check_done(Name,Flag)	Inquires about the status of the 69709A card's operation. If the card has completed, the value of Flag that is returned is a "1". If the card has not completed, the value returned is a "0". Once a value of "1" is returned, the Check_done statement will subsequently return a zero until the function is started again and completes another operation.
Enable_output(Name)	Allows the power supply to output the values which are programmed or have previously been programmed by the 69709A card.
Disable_output(Name)	Inhibits the controlled power supply's output and sets the power supply to a "safe" output state (zero volts).
Set_ovp_trip(Name)	Sets the over-voltage protection circuit of the power supply. As a result the output of the supply will drop to zero.
Reset_ovp_trip(Name)	Resets the over-voltage protection circuit of the power supply. The power supply will return to its previously programmed voltage and current.

Advanced Programming (continued)

Statement	Description
Set_local_vprog (Name,Flag)	Changes the output voltage programming source of the power supply from Remote (the 69709A card) to local (power supply's front panel). Make sure that the front panel controls are properly set before you switch over to local control. The Set statement enables local voltage control when the "Flag" variable is set to a one. If the "Flag" is set to a zero, control is changed back to Remote.
Set_local_iprog (Name,Flag)	Changes the output current programming source of the power supply from Remote (the 69709A card) to local (power supply's front panel). Make sure that the front panel controls are properly set before you switch over to local control. The Set statement enables local current control when the "Flag" variable is set to a one. If the "Flag" is set to a zero, control is changed back to Remote.

The following Set statements can be used in your program to enable and disable various interrupts that are available on the 69709A card. The interrupts can be used to signal the computer via the interrupt system when various conditions occur in the controlled power supply. The Set statement enables the applicable interrupt when the "Flag" variable is set to a one. If the "Flag" is set to a zero, the interrupt is disabled; i.e. it cannot signal the computer. In any case, remember to program an Enable_intr statement in order to allow all the selected "Set...ien" statements to be able to interrupt the computer.

Set_const_v_ien (Name,Flag)	Allows card to signal the computer when the power supply's output is being voltage regulated
Set_const_i_ien (Name,Flag)	Allows card to signal the computer when the power supply's output is being current regulated.
Set_unreg_ien (Name,Flag)	Allows card to signal the computer when the power supply's output is not controlled. This occurs when the supply is neither voltage regulated nor current regulated.
Set_over_v_ien (Name,Flag)	Allows card to signal the computer when the power supply's overvoltage protection circuitry has been activated.
Set_over_t_ien (Name,Flag)	Allows card to signal the computer when the temperature inside the power supply exceeds a safe level.
Set_acfault_ien (Name,Flag)	Allows card to signal the computer if the ac line voltage to the power supply falls below the minimum level for proper operation or the internal bias voltages are too low.

Advanced Programming (continued)

Statement	Description
Set_update_ien (Name,Flag)	Allows card to signal the computer whenever the output voltage and output current readings are updated by the 69709A card. This will occur typically every 100 milliseconds.
Set_end_cyc_ien (Name,Flag)	Allows card to signal the computer whenever it has completed its operation (sent a new interrupt only if it had previously been enabled).

When an interrupt occurs,the cause of the interrupt is passed to your interrupt service routine "User_handler" via the "Reason" parameter. Multiple interrupt causes can occur at the same time, so each possible cause is assigned a unique binary-weighted value as shown below. You must test for these values in your User_handler. Note that a value will appear as the cause of an interrupt only if it had previously been enabled.

Cause of Interrupt	Value
Constant Voltage	1
Constant Current	2
Unregulated Output	4
Over Voltage	8
Over Temperature	16
AC Fault	32
Readback Update	64
End of Cycle	128

The following functions check the status of the various power supply operating conditions. The value returned is a one if the condition is active; it is zero if the condition is not active. These functions can be used to programatically determine the state of the power supply without using the computer's interrupt system.

Statement	Description
FNConstvmode(Name)	Returns the status of the Constant Voltage(CV) mode flag. If the power supply is operating in the CV mode, a value of one is returned.
FNConstimode (Name)	Returns the status of the Constant Current(CC) mode flag. If the power supply is operating in the CC mode, a value of one is returned.
FNUnregout(Name)	Returns power supply's regulation status. If unregulated, a value of one is returned.
FNOvptrip(Name)	Returns power supply's Over Voltage Protection circuit status. If the circuit has tripped, a value of one is returned.

Advanced Programming (continued)

Statement	Description
FNOvertemp(Name)	Returns power supply's temperature status. If the temperature inside the power supply exceeds a safe level, a value of one is returned.
FNAcfault(Name)	Returns power supply's ac input status. If an ac line fault has been detected, a value of one is returned.
Set_timeout (Name,Value)	Changes the value of Timeout from the time specified in the menu to Value. Time is specified in seconds, from 0.001 to 128,000 seconds.

Power Supply Programming Using Interrupts

An application using programmable power supplies is component burn-in. Components are placed in a chamber at an elevated temperature. Power is applied to the components for an extended period of time. The purpose is to find weak components.

In the following example, three power supplies are used in a burn-in test. The interrupt system will be utilized so that if a component fails, the computer will be notified. This is accomplished by enabling the Constant Current Interrupt on each power supply. The power supplies will normally be in the Constant Votlage mode so that a Constant Current Interrupt will signal a failure. In this example, all three power supplies are initialized using the System_init statement.

```
10  COM/Names/Ps_1,Ps_2,Ps_3
20  Configure("Ask_me","Myfile")
30  !
40  ! Initialize the three power supplies
50  ! to Constant Voltage Mode, 0 volts output,
60  ! and Output Sampling ON
70  !
80  System_init
90  !
100 ! The output voltages of the three supplies
110 ! are programmed using Output statements
120 !
130 Output_voltage(Ps_1,5.00)
140 Output_voltage(Ps_2,10.00)
150 Output_voltage(Ps_3,20.00)
160 !
170 ! The Constant Current interrupts are enabled
180 ! on each power supply
190 !
200 Set_const_i_ien(Ps_1,1)
210 Set_const_i_ien(Ps_2,1)
220 Set_const_i_ien(Ps_3,1)
230 !
240 ! The interrupts from the power supplies
250 ! are enabled. A different vector number is used
260 ! so that each power supply can have its own
270 ! interrupt handling routine
280 !
290 Enable_intr(Ps_1,1)
300 Enable_intr(Ps_2,2)
310 Enable_intr(Ps_3,3)
320 !
330 ! Refer to Chapter 10 for Interrupt Handling
340 ! examples
      .
      .
      .
999 END
```

69720A/69721A Digital-to-Analog Converter Cards

Model 69720A is a twelve-bit digital-to-analog (D/A) voltage converter which can be programmed to output a voltage within the range from -10.240 to +10.235 volts with a minimum step change of 0.005 volts. It has the capability of supplying up to 5.0 milliamps of output current.

Model 69721A is a twelve-bit D/A current converter which provides bipolar current output from −20.48 to +20.47 milliamps with a compliance voltage of 11 volts and an LSB value of 10 milliamps. A voltage output is also available which tracks the current output. Refer to Operating Manual, HP Part No. 5950-1955, for complete specification and hardware details for the 69720A and 69721A D/A Converter cards.

Menu Entries for the D/A Converter Cards

The configuration and initial conditions for the D/A Converter cards are specified on a two page menu. Before programming either the 69720A or 69721A card, you must specify various configuration parameters in the menu for the applicable card. Instructions on how to use the menus and additional information on most of the parameters are given in Chapter 3. The menus for the two cards are identical except for the Model identifier in the menu heading. The following discussion uses the 69720A Voltage D/A menu as an example in order to describe the various parameters.

```
Name --- Volts_dac                Date: 21 Aug 1984
Model 69720A Voltage D/A          Time: 17:21:22
File Name -- Chapter_6
User ID ----
Time-stamp - First Time

                                  Page 1 of 2

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot ..........



Timeout ....... 5
```

Menu Page 1 of 2

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ..... 8
HP-IB Address ......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Menu Page 1 of 2 (continued)

Frame

Specifies the Multiprogrammer frame which contains the D/A Converter card. It has a legal range from 0 to 7. The default value is 0.

Slot

Identifies the location of the D/A card within the frame. The legal range is from 0 to 15. This parameter **MUST** be filled in; it has no default value.

Timeout

Specifies the time in seconds that will be allowed for the D/A card to complete its operation; that is, signal when its output has changed in response to a programmed value. If an Output or Wait_for statement does not complete within this time period, Error code 803 will be generated. The legal range for timeout is from 0.001 to 128,000 seconds. The default value is 5 seconds. Since the D/A card completes in 6 microseconds, the Timeout should never occur if you program a D/A card with an Output statement.

```
Name --- Volts_dac                    Date: 21 Aug 1984
Model 69720A Voltage D/A              Time: 17:21:44
File Name -- Chapter_6
User ID ----
Time-stamp - First Time

                                  Page 2 of 2


Data Conversion ..... Standard


Run Time Limit Checking ... No


Initial Value ..... 0
```

Menu Page 2 of 2

Data Conversion

Specifies how the data value sent to the D/A Converter card by an Output or Preset statement is to be scaled. One of three options can be selected: Standard, User, or None.

1. "Standard" conversion is the default for a D/A Converter card. For the 69720A card (−10.240 V to +10.235 V output range), the data values sent to the card represent volts. A value of −4.35 sent to the 69720A card would result in an output of −4.35 volts. For the 69721A card (−20.48 to +20.47 milliamps output range), the data values sent to the card represent amps. A value of .01275 sent to the 69721A card would result in an output of .01275 amps (or 12.75 milliamps).

2. "User" conversion allows the card data to be specified in terms of the application rather than in terms of the D/A card's standard units (volts or mA). As described below, four additional entries appear on the menu when User conversion is selected.

User's
P1 P2

Standard
P1 P2

Appear on the menu only when User conversion is selected. These entires allow you to fill in the ratio of User and Standard values so that the 14752A can convert the user defined data values into values which are recognized by the D/A Card. The P1 default values are 0; the P2 default values are 1.

Example: Assume that a hydraulic actuator requires a dc control voltage. The dc control voltage is supplied by the 69720A Voltage D/A Converter. At the "rest" position of 0 centimeters of the actuator, a dc voltage of 1.0 V is required; at its full extension of 10 centimeters, 8.75 V is required. Assume that a linear relationship exists between voltage and position. Using the "Standard" conver-

Menu Page 2 of 2 (continued)

sion option the card would be programmed in volts. In order to program the card in centimeters instead of in volts, select the "User" data conversion option and enter the following in the P1 and P2 parameters:

	P1 Entry	P2 Entry
User's	0 cm	10.00 cm
Standard	1.00 volts	8.75 volts

3. "None" conversion specifies that the data sent to the D/A card represents the number of LSB's (minimum programmable output) for the card. For D/A converter cards, the range for this data is from −2048 to +2047. The Output and Preset statements recognize the negative numbers as 2's complement binary values and the positive numbers as unsigned binary values for the D/A card. For the 69720A Voltage D/A (LSB = .005V), the values from −2048 to +2047 represent an output voltage range from −10.240V to +10.235V. For the 69721A Current D/A (LSB = .00001A), the values from −2048 to +2047 represent an output current range from -.02048A to +.02047A (or from −20.48mA to +20.47mA).

The "None" conversion option is the fastest way of sending data to a D/A card. An Output statement will execute 20 to 30% faster with the "None" conversion than with the "Standard" conversion; however, the "Standard" or "User" conversion provides an easier way of sending data to the card.

Run Time Limit Checking — If Yes is selected, the values sent to a D/A card will be checked against the High and Low Limits that you specify. The default for this parameter is off (No).

High Limit — Appears only when Run Time Limit Checking is on (Yes). Specifies the upper bound for the data value. The value entered must be greater than the Low Limit entered. Error #802 will be generated when a programmed value is greater than the High Limit entered. This value **MUST** be entered in the same units as was selected under Data Conversion. This parameter **MUST** be filled in; it has no default value.

Low Limit — Appears only when Run Time Limit Checking is on (Yes). Specifies the lower bound for the data value. The value entered must be lower than the High Limit entered. Error #802 will be generated when a programmed value is less than the Low Limit entered. This value **MUST** be entered in the same units as was selected under Data Conversion. This parameter **MUST** be filled in; it has no default value.

Initial Value — Specifies the value which the D/A card will output when it is initialized by an Init or System_init statement. This value **MUST** be entered in the same units as was selected under Data Conversion. The default value is 0 standard unit (0 volts for the 69720A and 0 amps for the 69721A).

Programming Fundamentals

Only two 14752A statements are required to program a D/A Converter card. Each statement is described below and then a simple example program is given. For details concerning the syntax requirements for these statements, refer to Chapter 3.

Statement	Description
Init(Name)	Sets the output of the named D/A card to the Initial Value specified in the menu. Init also clears the EOP signal at the card's edge connector.
Output(Name,Value)	Sets the output of the named D/A card to the value specified in the variable "Value".

Resistor Testing Programming Example

The 69721A Current D/A Converter card can be used to supply a precision programmable output current to measure the value of resistors.

The following example uses a 69721A card to supply a precision current through the resistor and a DVM is used to measure voltage across the resistor. The resistor value equals the measured voltage across the resistor divided by the programmed current through the resistor.

In this example, the "Standard" conversion option is used. A value of .001 will result in an output current of + 1.00 milliamps. The 69721A Current D/A is identified with the Name "Current_source". During the configuration phase, the Name is defined, the Initial Output is set to 0, and "Standard" Data Conversion option is selected.

```
10  COM/Names/Current_source
20  Configure("Ask_me","MYFILE")
30  !
40  ! Initialize the 69721A for an output
50  ! current of 0 milliamps to define
60  ! a safe state for the test.
70  !
80   Init(Current_source)
90  !
100 ! Output a current value of 1.00 milliamps(.001A)
110 ! to test a resistor and print a message
120 ! to notify the operator.
130 !
140   Output(Current_source,.001)
150 !
160 PRINT "Current set to 1.00 milliamps."
170 END
```

Advanced Programming

There are 14752A statements which allow special purpose programming of the 69720A and 69721A D/A Converter cards. They are considered advanced programming statements because you need a good understanding of the Multiprogrammer System in order to use them in a meaningful program. In general, these statements are used only when precise control of the card's output operation is a concern. They allow features such as interrupts and parallel operation of multiple cards. All of these features are described in Chapter 10. The advanced programming statements that can be used to program a D/A Converter card are described as follows:

Statement	Description
Translate(Name, Iarray(*),User(*))	Used in conjunction with Output_iblock to process data for the D/A card according to the specified data conversion. Data is taken from the REAL array named "User", converted to Integer numbers, and returned into the INTEGER array named "Iarray". All arrays **MUST** have been previously DIMensioned or ALLOCATEd in the user's program otherwise a BASIC system error (#13) will be generated.
Output_iblock(Name, Iarray(*)[,Count[,Index]]	Used to send integer data from the array named "Iarray directy to the D/A card. Since this array is the same array as "Iarray" used in the Translate statement, the array names are the same. The number of words sent to the card is equal to the size of the data array. The optional parameter Count, specifies the number of words that will be sent to the card. This number must be less than, or equal to, the size of the array. Along with Count, Index can be used to specify the array location of the first data item that will be sent. The data array **MUST** have been previously DIMensioned or ALLOCATEd in the user's program otherwise a BASIC system error (#13) will be generated. This statement presets the card and waits for an external trigger. Therefore, in order to use Output_iblock, the D/A card must be triggered externally.
Preset(Name,Value)	Sends a data value to the named card but does not cause the output to change. The Preset statement is used as a setup operation for the Start statement or for an external trigger signal input.
Start(Name1,..., Name8)	Sets the output of the named card(s) to the last value sent to the card (e.g. by a Preset statement). If more than one name is listed in the Start statement, the outputs on all named cards are changed simultaneously and any "Named" input cards are started.
Wait_for(Name)	Suspends program execution until the D/A card has completed operation or a timeout has occurred. Used if you want to wait for the card to finish when it has been started by an External Trigger signal or Start statement. The Timeout parameter in the menu specifies the timeout period. Error #803 is generated if Timeout occurs before the card completes operations.

Advanced Programming (continued)

Statement	Description
Check_done(Name,Flag)	Inquires about the status of the D/A card's operation. If the card has completed, the value of Flag that's returned is a "1". If the card has not completed, the value returned in a "0". Once a value of "1" is returned, Check_done will subsequently return a "0" until the card is started again and it completes another operation (outputs another value).
Enable_intr(Name,Vector)	Allows the D/A Converter card to signal the computer via the interrupt system when it has finished operations. This statement must be used in conjunction with the BASIC statement ON INTR as described in Chapter 10.
Disable_intr(Name)	Inhibits the D/A Converter card from signalling the computer when it has completed operations. It is meaningful only when an Enable_intr statement has already been sent. Note that the Init and System_init statements will also cancel a previously issued Enable_intr statement.
Set_timeout(Name,Value)	Changes the value of Timeout from the time specified in the menu to Value. Time is specified in seconds, from 0.001 to 128,000 seconds.

Advanced Programming Example

Assume that a 69720A Voltage D/A Converter card is used to activate a mechanical actuator. The D/A conversion is to be started by an external signal applied to the card's external trigger input (see Chapter 10). In this example, a Preset statement will be used to set the required output level. A Wait_for statement is used to suspend the program until the D/A card has been activated by the external signal. The D/A card is named "Arm_control".

```
10  COM/Names/Arm_control
20  Configure("Ask_me","MYFILE")
30  !
40  ! The D/A card is initialized to 0 Volts
50  ! output, Standard Data Conversion,
60  ! and interrupts disabled.
70  !
80  Init(Arm_control)
90  !
100 ! Preset the D/A for a 10.5V output
110 !
120 Preset(Arm_control,10.5)
130 !
140 ! Suspend program execution until the
150 ! external trigger signal is received, or
160 ! until Timeout occurs.
170 !
180 Wait_for(Arm_control)
190 !
200 ! At this point, the D/A output has been
210 ! initiated by the external trigger and
220 ! program execution will continue. A message
230 ! is printed to flag the operator.
240 !
250 PRINT "Arm Control Activated"
260 END
```

Chapter 7
Digital Input Functions

Introduction

Electrical signals can be classified as being either analog or digital. An analog signal is characterized by being continuous in nature, assuming a value within a range of possible values. A digital signal, on the other hand, is characterized as being discrete in nature. The signal can assume either one of two values; these values are usually referred to as "states". Digital signals can be sent along a single signal line or multiple signal lines. Multiple signal lines are commonly referred to as a digital "bus". The Digital Input Functions of the Multiprogrammer read digital signals from either single signal lines or a digital "bus" of up to sixteen signal lines.

The Multiprogrammer provides digital input functions in the form of the 69770A Isolated Digital Input card and the 69771A Digital Input/Analog Comparator card.

69770A Isolated Digital Input Card

The 69770A Isolated Digital Input card monitors the logic states of up to 16 digital signal lines that can be biased up to 250 volts ac or dc above earth ground or that must remain isolated from earth ground for reasons of safety or noise immunity. An optically coupled isolator in each of the card's 16 digital input lines breaks the path of a potential ground loop.

The outputs of the card's optically coupled isolators are brought out to its edge connector so that this card can be used as an input isolator for a 69776A Interrupt card, a 69790B Memory card/69791A Memory Card System, or another input device.

The card also has control signals at the edge connector which allow external devices (eg. other I/O cards) to "handshake" data into the card. Two of these signals are of particular importance: GATE and FLAG. The GATE signal is generated by the card whenever it must read an input word. The FLAG is an input signal returned to the card by the external device, to tell the card the data is ready to be read. Upon receiving the FLAG signal the card will read the data. The Isolated Digital Input card is shipped from the factory with the GATE signal wired to the FLAG signal by jumper W6 on the card. Thus, the 69770A is always ready to read the data at its input. However, when jumper W6 is removed the card will not complete the input operation until the FLAG signal is received and the data is read. If your application requires handshaking with the 69770A, you should read the "Front-plane Signals and Timing" section in Chapter 10.

More information (card input pins, control signals, etc.) on the model 69770A is contained in its Operating Manual (HP part number 69770-90003).

Menu Entries for Model 69770A

Before programming the 69770A Isolated Digital Input card, you must specify various parameters in the two page configuration menu for the card. Following are descriptions of all the parameters required to configure this function. Instructions on how to use the menus and additional information on most of the parameters are given in Chapter 3. The Isolated Digital Input function's parameters are described as follows:

```
Name --- Iso_dig_in              Date: 21 Aug 1984
Model 69770A Iso. Digital In     Time: 17:23:48
File Name -- Chapter_7
User ID ----
Time-stamp - First Time

                                 Page 1 of 2

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot ..........
Length ........ 16
Start Bit ..... 0


Timeout ....... 5
```

Menu Page 1 of 2

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ..... 8
HP-IB Address ......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the Isolated Digital Input card. The legal range is from 0 to 7. The default value is 0.

Menu Page 1 of 2 (continued)

Slot

Identifies the location of the Isolated Digital Input within the frame. It has a legal range from 0 to 15. This parameter **MUST** be filled in; it has no default value.

Start Bit

Specifies the bit within the 16-bit binary input word that is to be used to begin the group of bits referenced by the "Name" identifier. The legal range is from 0 to 15. The default value is 0.

Length

Specifies the total number of binary bits, beginning with the Start Bit, in the group. The legal range of values is from 1 to 16. The default value is 16.

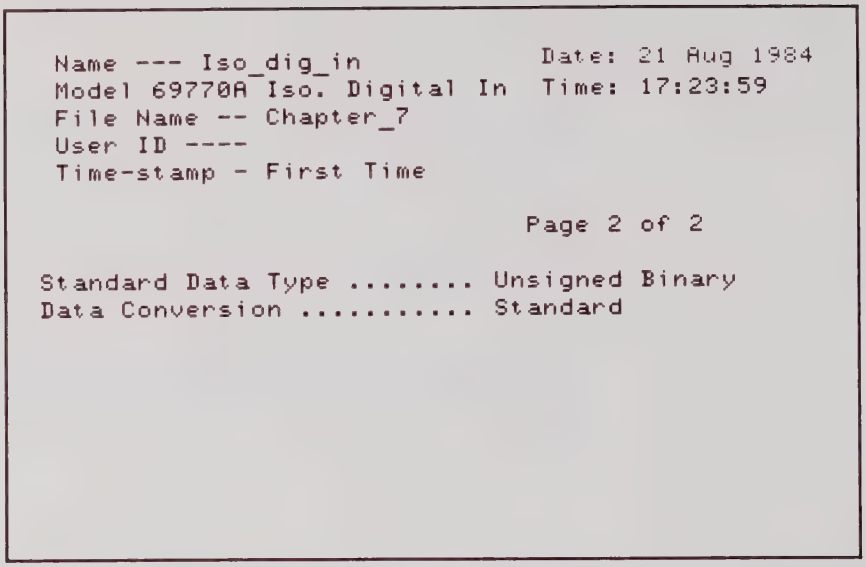
Examples:

The 16 bits of a Digital Input card are designated 0 (LSB) through 15 (MSB). Here are three examples of entering Start Bit and Length parameters to associate the desired bits with a name. Only the Named bits will be returned by the Input and Get statements.

1. To associate all 16 bits with a name, Start Bit = 0 Length = 16.
2. To associate bits 6 through 8 with a name, Start Bit = 6 Length = 3.
3. To associate bit 10 with a name, Start Bit = 10 Length = 1.

Timeout

Sets the time in seconds that will be allowed for the Isolated Digital Input card to complete its operation. If the Input or Wait_for statements do not detect completion within this time, a Timeout error message (error code 803) is generated. Any number between 0.001 and 128,000 may be entered here. The default value is 5 seconds and normally would not be changed unless you have cut jumper W6 on the card and connected the Gate and Flag control lines to some external device.



Menu Page 2 of 2

Menu Page 2 of 2 (continued)

Standard Data Type

Specifies either "Unsigned Binary" or "2's Complement Binary" as the Standard Data Type for the card. The default selection is Unsigned Binary.

1. Unsigned Binary. The 14752A Input and Get statements interpret the numbers read by the card as unsigned binary values. The range of numbers that can be read is from 0 to $2^N - 1$; where N is the number of "Named" bits specified by the Start Bit and Length parameters.

Examples:

a. If all 16 bits, 0(LSB) to 15(MSB), are associated with a name, the range is from 0 to $2^{16} - 1$, or from 0 (all 16 bits off) to 65535 (all 16 bits on).

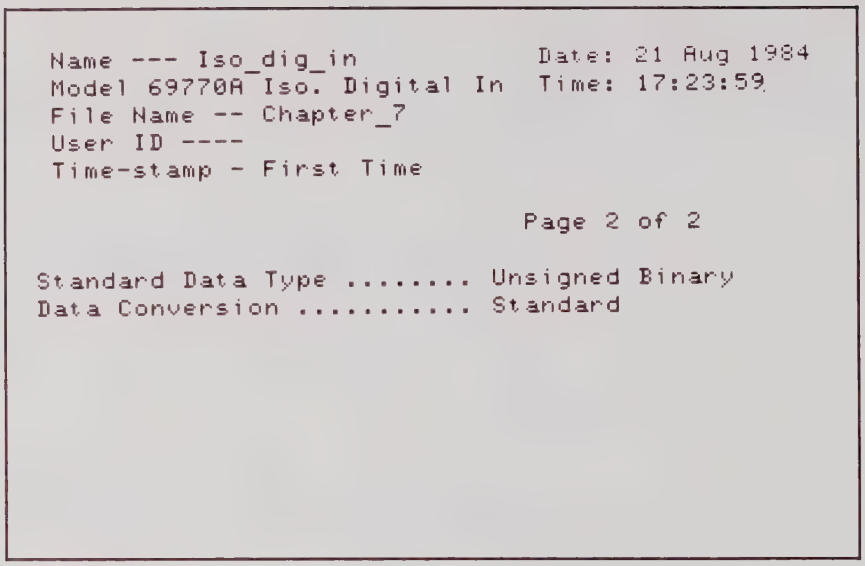
b. If any 4 consecutive bits (e.g. bits 4 to 7) are associated with a name, the range is from 0 to $2^4 - 1$, or from 0 (all 4 bits off) to 15 (all 4 bits on).

2. 2's Complement Binary. The 14752A Input and Get statements interpret input data as 2's complement binary values for the Digital input card. The range of numbers that are returned is from $-2^{(N-1)}$ to $+[2^{(N-1)}] - 1$ were N is the number of "Named" bits.

Examples:

a. If all 16 bits, 0(LSB) to 15(MSB), are associated with a name, the range is from -2^{15} to $+2^{15} - 1$, or from -32768 (all bits off except bit 15) to 32767 (all bits on except bit 15).

b. If any 4 consecutive bits (e.g. 4 to 7) are associated with a name, the range is from -2^3 to $+2^3 - 1$, or from -8 (all bits off except bit 7) to 7 (all 4 bits on except bit 7).



Menu Page 2 of 2

Menu Page 2 of 2 (continued)

Data Conversion

Specifies how the data value is to be scaled by the Input and Get statements. One of two data conversion options can be selected: Standard and User.

1. "Standard" conversion for this card means that the data values returned from the card are not scaled and are merely returned as numbers in the form of 2's complement or unsigned binary, whichever was selected as the standard data type. Standard is the default data conversion for this function and, if the data type chosen was unsigned binary, the values returned will be from 0 to 65535.

2. "User" conversion allows data to be specified in terms of the application rather than in terms of the card's standard units. This means you can have the data scaled for you when it is returned to the computer. As shown below, four additional entries appear on the menu when User conversion is selected.

User's
P1 P2

Standard
P1 P2

Are values used only when the User conversion option is selected. These parameters allow you to fill in the ratio of User units and Standard units values so the 14752A can convert the data returned by the card into user defined units. The P1 default values are 0; P2 default values are 1.

More About Data Conversion

Standard conversion is used when you want to read data from a card in the units it is accustomed to. In other words, Standard units are the fundamental units of a function, a positive 16 bit number in our case (provided Start Bit = 0 and Length = 16). Standard Units are also the fastest way to get data from this function. An Input or Get statement will execute 20% to 30% faster with Standard units than with User units.

User Units give us a much easier way to get data from a function, but it takes the 14752A software 20% to 30% longer to process the data. When User conversion is selected in the menu, four other parameters must also be specified. These parameters give the 14752A all the data it needs to perform the necessary conversion calculations. An example which illustrates how programming units can be helpful when reading data from a digital input card turn to the paragraph titled "More About Data Conversion" in the section for the 69771A card. More information concerning this subject is also available in Chapter 3.

Programming Fundamentals

Only two fundamental statements are needed to program the Isolated Digital Input card. Each statement is described below and then a simple example program is given. For details concerning the syntax requirements for these statements, refer to Chapter 3.

Statement	Description
Init(Name)	Sets the card to the initial power-up state. The card's output control signals (described in Chapter 10) are cleared, Interrupts are disabled, and the value of Timeout is reset in the time specified in the menu.
Input(Name,Value)	Reads the data value at the input of the 69770A card and sends it to the controller. This statement will not complete until either FLAG is returned or a timeout occurs. Note: since GATE is jumpered to FLAG with jumper W6, the statement will complete immediately.

Example 1. Using the 69770A

It's very simple to read data from the Isolated Digital Input card. Suppose we had an instrument which has a 16 bit two's complement TTL output that was intended for a 16 bit computer interface and we wanted to read the instrument's output with an Isolated Digital Input card. First we would have to change the Standard Data Type to two's complement on menu page 2. Then the following sample program will read the data into our Series 200 computer and print the result.

```
10 COM/Names/Instrument
20 Configure("Menu_on","MY_FILE")
30 Init(Instrument)
40 Input(Instrument,Value)
50 PRINT Value
60 END
```

Since the instrument and the 69770A are both two's complement, the Value which is printed will be within the range -32768 to +32767. If the instrument was a meter that measured voltages between -32.768 and +32.767 Volts with a resolution of one millivolt, then you would have to change the Data Conversion menu parameter to User and make the following enteries:

	P1	P2
User's	−32.768	32.767
Standard	−32768	32767

The same programming sample given above would therefore print the results scaled in terms of Volts.

Advanced Programming

There are 14752A statements which allow special purpose programming of the Isolated Digital Input function. They are considered advanced programming statements because you need a good understanding of the Multiprogrammer I/O cards and system to put together a meaningful program. In general, these statements are used only when precise control of the timing is a concern. They allow features such as parallel inputs from more than one function card, external triggering, checking status, and interrupts.

Statement	Description
Input_iblock(Name, Iarray(•)[,Count[,Index]])	Takes a block of readings from the Digital Input card and stores the data into the INTEGER array called "Iarray". The number of words read from the card is equal to the size of the data array. The optional parameter Count, specifies the number of words that will be read from the card. This number must be less than, or equal to, the size of the array. Along with Count, Index can be used to specify the array location where the first data item will be placed. The data array must have been previously DIMensioned or ALLOCATEd otherwise a BASIC system error (#13) will be generated. This statement starts the card and waits for the Gate/Flag handshake to complete (see Chapter 10 for details on Gate/Flag handshake).

The Translate statement described below allows the user to separate the data conversion process from the measurement process. This capability is provided by combining this statement with Input_iblock. The Input_iblock statement is used to quickly collect raw integer data from the Digital Input card. The Translate statement can then be used to process this data into real numbers using the Digital Input card's specified data conversion.

Translate (Name, Iarray(•),User(•))	Used after Input_iblock to process the data of the Digital Input card. The Digital Input card is specified by the parameter, Name. The array named "Iarray" is an array of integers. Since this array is the same array as "Iarray" used in Input_iblock, the array names are the same. The integer data is taken from "Iarray" and converted to real numbers. The data is then returned into the REAL array named "User" . All arrays must have been previously DIMensioned or ALLOCATEd in the user's program otherwise a BASIC system error (#13) will be generated.
Start(Name1,...,Name8)	Makes the card take a reading. Used as a set-up operation for a Get statement. If more than one Name is listed, all the named functions read their data at the same instant, in parallel.
Wait_for(Name)	Program execution is paused until the input data has been read or a timeout occurs. Used if you want to wait for the Isolated Digital Input card to finish when it has been started with a Start statement or an External Trigger. Especially useful when using the Gate/Flag handshake.

Advanced Programming (continued)

Check_done(Name,Flag) Inquires about the status of the function's operation. Flag is set to a one if the card has completed. If the card has not completed, the value returned is a "0". Once a value of one is returned, the Check_done statement will subsequently return zero until the function is started again.

Get(Name,Value) Returns a data reading from the card's storage register after it has been placed there via a Start statement or an External Trigger.

Set_timeout(Name,Value) Changes the value of Timeout from the time specified in the menu to Value. Time is specified in seconds, from 0.001 to 128,000 seconds.

Enable_intr(Name,Vector) Allows the Isolated Digital Input card to signal the controller via the interrupt system when it has finished reading the data value at its edge connecter. The statement must be used in conjunction with the BASIC statement ON INTR as described in Chapter 10 of this manual. The Vector variable identifies the function that caused the interrupt. Note, if this statement is used, the Init and System_init statements will cause the interrupts to become disabled as if a Disable_intr statement was executed.

Disable_intr(Name) Inhibits the Isolated Digital Input card from signaling the controller when it's done. It is meaningful only after the Enable_intr statement has been programmed.

69771A Digital Input/Analog Comparator Card

The Digital Input/Analog Comparator card monitors the logic states of up to 16 switch contacts, digital signals, or analog signals that are referenced to data common. As shipped, the card's analog comparator inputs are internally biased at + 1.2 volts, a bias suitable for TTL input levels. A closed contact to ground, a TTL low, or an analog level less positive than the bias voltage is interpreted as a logic 0, and their opposites are interpreted as a logic 1. This logic sense can be inverted for all 16 inputs by installing one jumper (W48) on the card.

The analog comparators at the card's inputs have their outputs available at the card edge connector so that this card can be used as a signal conditioner for other input devices or plug-in cards such as the 69776A Interrupt card.

The card also has control signals at the edge connector which allow external devices (eg. other I/O cards) to "handshake" data into the card. Two of these signals are of particular importance: GATE and FLAG. The GATE signal is generated by the card whenever it must read an input word. The FLAG is an input signal returned to the card by the external device, to tell the card the data is ready to be read. Upon receiving the FLAG signal the card will read the data. The Digital Input/Analog Comparator card is shipped from the factory with the GATE signal wired to the FLAG signal by jumper W52 on the card. Thus, the 69771A is always ready to read the data at its input. However, when jumper W52 is removed, the card will not complete until the FLAG signal is received and the data is read. If your application requires handshaking with the 69771A, you should read the "Frontplane Signals and Timing" section in Chapter 10.

More information (card input pins, control signals, etc.) on the model 69771A is contained in its Operating Manual (HP part number 69771-90003).

Menu Entries For Model 69771A

Before programming the 69771A Digital Input/Analog Comparator card, you must specify various parameters in the two page Configuration Menu for the card. Following are descriptions of all the parameters required to configure this function. The parameters on the menus, as well as how to use the menu keys, are described in detail in Chapter 3.

```
Name --- Comp_dig_in              Date: 21 Aug 1984
Model 69771A Comp. Digital In Time: 17:25:15
File Name -- Chapter_7
User ID ----
Time-stamp - First Time

                                  Page 1 of 2

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot ..........
Length ........ 16
Start Bit ..... 0


Timeout ....... 5
```

Menu Page 1 of 2

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ..... 8
HP-IB Address ......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the Digital Input/Analog Comparator card. It has a legal range for 0 to 7. The default value is 0.

Slot

Identifies the location of the Digital Input/Analog Comparator card within the frame. It has a legal range from 0 to 15. This parameter **MUST** be filled in; it has no default.

Menu Page 1 of 2 (continued)

Start Bit

Specifies the first bit within a group of bits on a 69771A card. It has a legal range from 0 to 15; the default value is 0.

Length

Specifies the total number of binary bits, beginning with the Start Bit, in the group. The legal range of values is from 1 to 16; the default is 16.

Examples:

The 16 bits on a Digital Input card are designated Bit 0 (LSB) through Bit 15 (MSB). Here are three examples of entering Start Bit and length parameters to associate the desired bit numbers with a name. Only the "named" bits will be returned by the Input and Get statements.

1. To associate all 16 bits with a name, Start Bit = 0 Length = 16.

2. To associate bits 3 through 5 with a name, Start Bit = 3 Length = 3.

3. To associate bit 10 with a name, Start Bit = 10 Length = 1.

Timeout

Sets the amount of time (in seconds) that will be allowed for the Digital Input/Analog Comparator card to complete its operation. If the Input or Wait_for statements do not detect completion within this time, a Timeout error message (Error code 803) is generated. Any number between 0.001 and 128,000 may be entered for Timeout. The default value is 5 seconds and normally would not be changed unless you have cut jumper W52 on the card and connected the Gate and Flag control lines to some external device.

```
Name --- Comp_dig_in              Date: 21 Aug 1984
Model 69771A Comp. Digital In Time: 17:25:50
File Name -- Chapter_7
User ID ----
Time-stamp - First Time

                                Page 2 of 2

Standard Data Type ..... Unsigned Binary
Data Conversion ........... Standard
```

Menu Page 2 of 2

Standard Data Type

Specifies either "Unsigned Binary" or "2's Complement Binary" as the Standard Data Type for the card. The default selection is Unsigned Binary.

1. Unsigned Binary. The 14752A Input and Get statements interpret the numbers read by the card as unsigned binary values. The range of numbers that can be read is from 0 to $(2^N - 1)$; where N is the number of "Named" bits specified by the Start Bit and Length parameters.

Examples:

a. If all 16 bits, 0(LSB) to 15(MSB), are associated with a name, the range is from 0 to $2^{16} - 1$, or from 0 (all 16 bits off) to 65535 (all 16 bits on).

b. If any 4 consecutive bits (e.g., bits 4 to 7) are associated with a name, the range is from 0 to $2^4 - 1$, or from 0 (all 4 bits off) to 15 (all 4 bits on).

2. 2's Complement Binary. The 14752A Input and Get statements interpret input data as 2's complement binary values for the Digital input card. The range of numbers that are returned is from $-2^{(N-1)}$ to $+[2^{(N-1)}] - 1$; where N is the number of "Named" bits.

Examples:

a. If all 16 bits, 0(LSB) to 15(MSB), are associated with a name, the range is from -2^{15} to $(+2^{15}) - 1$, or from -32768 (all bits off except bit 15) to 32767 (all bits on except bit 15).

Menu Page 2 of 2 (continued)

b. If any 4 consecutive bits (e.g., 4 to 7) are associated with a name, the range is from -2^3 to $+2^3-1$, or from -8 (all bits off except bit 7) to 7 (all 4 bits on except bit 7).

Data Conversion

Specifies how the data value is to be scaled by the Input and Get statements. One of two data conversion options can be selected: Standard and User.

1. "Standard" conversion for this card really means that the data values returned from the card are not scaled and are merely returned as numbers in the form of 2's complement or unsigned binary, whichever was selected as the standard data type. Standard is the default data conversion for this function, and if the data type chosen was unsigned binary, the values returned will be from 0 to 65535.

2. "User" conversion allows data to be specified in terms of the application rather than in terms of the card's standard units. This means you can have the data scaled for you when it is returned to the computer. As shown below, four additional entries appear on the menu when User conversion is selected.

User's
P1 P2

Standard
P1 P2

These are values used only when the User conversion option is selected. These parameters allow you to fill in the ratio of User units and Standard units values so the 14752A can convert the Standard data returned by the card into user defined units. The P1 default values are 0; P2 default values are 1.

More About Data Conversion

Standard conversion is used when you want to read data from a card in the units it is accustomed to. In other words, Standard units are the fundamental units of a function, a positive 16 bit number in our case. Standard units are also the fastest way to get data from this function. An Input or Get statement will execute 20% to 30% faster with Standard units than with User units.

User units give us a much easier way to get data from a function, but it takes the 14752A software 20% to 30% longer to process the data. When User conversion is selected in the menu, four other parameters must also be specified. These parameters give the 14752A all the data it needs to perform the necessary conversion calculations.

Here is an example which illustrates how programming units can be helpful when reading data from a Digital Input/Analog Comparator card:

Suppose we have an 8-bit analog to digital converter that has a voltage input range from 0.0 Volts to 10.24 Volts. We know from reading the specification sheet for the A/D, that 0 Volts yields a binary output of 00000000, and 10.24 Volts will give a binary output of 11111111. Therefore, 0 Volts corresponds to 0 bits on and 10.24 Volts corresponds to all eight bits on (ie. the decimal number 255). To read any value from the card in terms of Volts, we simply enter the following menu parameters for User Conversion:

	P1	P2
User's	0	10.24
Standard	0	255

Programming Fundamentals

Only two fundamental statements are needed to program the Digital Input/Analog Comparator card. The following is a summary of these statements. For details concerning the syntax requirements for the Name and Value parameters, refer to Chapter 3.

Statement	Description
Init(Name)	Used to set the card to the initial power-up state. The card's output control signals (described in Chapter 10) are cleared, card interrupts are disabled,and the value of Timeout is reset to the time specified in the menu.
Input(Name,Value)	Reads the data value at the input of the 69771A card and sends it to the controller. This statement will not complete until either FLAG is returned or a timeout occurs. Note, if GATE is jumpered to FLAG with jumper W52 (the as shipped condition), the statement will complete immediately.

Programming Fundamentals (continued)

Example 1. Using the 69771A

This example shows how to read input data from a Digital Input/Analog Comparator card using the concept of defining more than one Name for a particular function. The 14752A programming package allows multiple names to be defined for each of these cards dividing the 16 channels into groups. The groups can be from 1 to 16 channels in width, and the channels must be adjacent. The 69771A card returns the full 16 bit word from the card, and then the 14752A masks off the bits which are not related to those defined for the group.

Let's assume that a Digital Input card has been connected to a three digit thumb-wheel switch or decade dial. Each digit of the switch goes from 0-9 and has BCD encoded outputs (each digit generates a 4 bit binary number from 0000 through 1001). The total input from the switch goes from 000 to 999. If the 12 input bits are read as a single 12 bit number, the resultant binary number will not correspond to the decimal reading on the switch. The correct reading must be calculated from the three individual digits of the switch. By using three different Names to define three groups of four bits each, the three digits of the switch can be read directly into three different variables, simplifying the the required calculation. The three Names defined could be Ones, Tens, and Hundreds. The Name "Ones" would have menu entries of 0 for start bit and 4 for length, corresponding to bits 0 through 3 on the card. The Name "Tens" would have menu entries of 4 for start bit, and 4 for length, corresponding to bits 4 through 7 on the card. The Name "Hundreds" would have menu entries of 8 for Start Bit and 4 for Length, corresponding to bits 8 through 11 on the card. The program looks like this:

```
10 COM /Names/Ones,Tens,Hundreds
20 Configure("Menu_on","SWITCH")
30 System_init
40 Input(Ones,C)
50 Input(Tens,B)
60 Input(Hundreds,A)
70 Result = 100*A + 10*B + C
80 PRINT Result
90 END
```

Since all three names are part of the same physical register, the decade dial should not be rotated until the three Input statements have executed. The Input statements read the three 4 bit fields into three variables, A, B, and C. The three variables are then used to reconstruct the correct result.

Programming Fundamentals (continued)

Example 2. Using the 69771A

Suppose we have two A/D converters like the one used in the previous example for Data Conversion. Let's read their digital outputs with one Digital Input card located in slot 5. We must use the menus to configure two different Names, one for each A/D. The outputs of the first A/D, named Converter1, will be connected to bits 0 through 7 of the Digital Input card, and the outputs of the second A/D, named Converter2, will be connected to bits 8 through 15. From the A/D's specifications we can fill in the menu parameters as follows:

Name: Converter1
Slot: 5
Start Bit: 0
Length: 8
Data Conversion: User
  Standard P1: 0 P2: 255
  User's P1: 0 P2: 10.24

Name: Converter2:
Slot 5:
Start Bit: 8
Length: 8
Data Conversion: User
Standard P1: 0 P2: 255
User's P1: 0 P2: 10.24

The menu parameters not mentioned above should be left set to their default values for this example. A simple program that would read the two A/D converters looks like this:

```
10 Com/Names/ Converter1,Converter2
20 Configure ("Menu__On","Test12")
30 System__init
40 Input (Converter1,Voltage1)
50 Input (Converter2,Voltage2)
60 Print "The first A/D measured "; Voltage1; "Volts."
70 Print "The second A/D measured "; Voltage2; "Volts."
80 END
```

Advanced Programming

There are 14752A statements which allow special purpose programming of the Digital Input/Analog Comparator function. They are considered advanced programming statements because you need a good understanding of the Multiprogrammer I/O cards and system to put together a meaningful program. In general, these statements are used only when precise control of the timing is a concern. They allow features such as parallel inputs from more than one function card, external triggering, checking status, and interrupts. All of these features are explained in greater detail in Chapter 10. The following is a list of the advanced programming statements which can be used for the 69771A Digital Input/Analog Comparator function.

Statement	Description
Input_iblock(Name, Iarray(•)[,Count[,Index]])	Takes a block of readings from the Digital Input card and stores the data into the INTEGER array called "Iarray". The number of words read from the card is equal to the size of the data array. The optional parameter Count, specifies the number of words that will be read from the card. This number must be less than, or equal to, the size of the array. Along with Count, Index can be used to specify the array location where the first data item will be placed. The data array must have been previously DIMensioned or ALLOCATEd otherwise a BASIC system error (#13) will be generated. This statement starts the card and waits for the Gate/Flag handshake to complete.

The Translate statement described below allows the user to separate the data conversion process from the measurement process. This capability is provided by combining this statement with Input_iblock. The Input_iblock statement is used to quickly collect raw integer data from the Digital Input card. The Translate statement can then be used to process this data into real numbers using the Digital Input card's specified data conversion.

Translate (Name, Iarray(•),User(•))	Used after Input_iblock to process the data of the Digital Input card. The Digital Input card is specified by the parameter, Name. The array named "Iarray" is an array of integers. Since this array is the same array as "Iarray" used in Input_iblock, the array names are the same. The integer data is taken from Iarray and converted to real numbers. The data is then returned into the REAL array named "User" . All arrays must have been previously DIMensioned or ALLOCATEd in the user's program otherwise a BASIC system error (#13) will be generated.
Start(Name1,..,Name8)	Makes the card take a reading. Used as a set-up operation for a Get statement. If more than one Name is listed, all the named functions read their data at the same instant, in parallel.
Wait_for(Name)	Program execution is paused until the input data has been read or a timeout occurs. Used if you want to wait for the Digital Input/Analog Comparator card to finish when it has been started with a Start statement or an External Trigger. Especially useful when using the Gate/Flag handshake.

Advanced Programming (continued)

Check__done(Name,Flag) Inquires about the status of the function's operation. Flag is set to a one if the card has completed. If the card has not completed, the value returned is a "0". Once a value of one is returned, the Check__done statement will subsequently return zero until the function is started again.

Get(Name,Value) Returns a data reading from the card's storage register after it has been placed there via a Start statement or an External Trigger.

Set__timeout(Name,Value) Changes the value of Timeout from the time specified in the menu to Value. Time is specified in seconds, from 0.001 to 128,000.

Enable__intr(Name,Vector) Allows the Digital Input/Analog Comparator card to signal the controller via the interrupt system when it has finished reading the data value at its edge connecter. Must be used in conjunction with the BASIC interrupt statement ON INTR as described in Chapter 10 of this manual. The Vector variable identifies the function that caused the interrupt. Note, if this statement is used, the Init and System__init statements will cause the interrupts to become disabled as if a Disable__intr statement was executed.

Disable__intr(Name) Inhibits the Digital Input/Analog Comparator card from signaling the controller after it is done. Meaningful only when the Enable__intr statement has been programmed.

Chapter 8
Digital Output Functions

Introduction

Many CAT systems require a digital stimulus or switching contacts to provide digital control of instruments, indicators, units under test, or test fixtures. Programmable switching contacts can also be used to connect analog signals to test instruments or units under test.

The Multiprogrammer provides two digital output functions. For those applications which require switch closures, the 69730A Relay Output card provides 16 separate relay contacts. A strictly digital stimulus is provided by the 69731B Digital Output card which contains 16 TTL-level outputs. The desired function is connected with the user's specified name using the Name Selection and Function Selection menus as described in Chapter 3.

The 14752A can program single or multiple outputs from the 69730A Relay Output and the 69731B Digital Output functions. This is accomplished by assigning Names to one output or any group of contiguous outputs on the applicable function. Entries on each function's menu specify which output or group of outputs are assigned to the Names(s). The following paragraphs describe the menus and 14752A statements used to program the 69730A Relay Output and the 69731B Digital Output functions.

69730A Relay Output Card

This card contains 16 independent, SPST, normally open, mercury wetted relay contacts. Because the relays require time to complete their operations, a built-in time delay of 6 milliseconds is included on the card to allow the relays time to settle. The relay contacts can be used as separate switches or can be externally connected to form a scanner or a matrix of switches. Refer to the 69730A Relay Output card Operating Manual, (HP Part No. 69730-90001), for complete hardware details.

Menu Entries for Model 69730A

The configuration and initial conditions for the 69730A Relay Output function are specified on a two page menu. The user is required to fill in the various parameters on the menu in order to program the function. Instructions on how to use the menus and additional information on most of the parameters are given in Chapter 3. The Relay Output function's parameters are described as follows:

```
Name --- Relay                    Date: 21 Aug 1984
Model 69730A Relay Actuator       Time: 17:28:07
File Name -- Chapter_8
User ID ----
Time-stamp - First Time

                                  Page 1 of 2

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot .......... 5
Start Bit ..... 0
Length ........ 16


Timeout ....... 5
```

Menu Page 1 of 2

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ..... 8
HP-IB Address ......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the Relay Output card. The legal range is from 0 to 7. The default value is 0.

Slot

Identifies the location of the Relay Output card within the frame. The legal range is from 0 to 15. This parameter **MUST** be filled in; it has no default value.

Menu Page 1 of 2 (continued)

Start Bit

Specifies the first contact within a group of contacts on the Relay Output card. The legal range is from 0 to 15. The default value is 0.

Length

Specifies the total number of contacts, beginning with the Start contact, in the group. The legal range is from 1 to 16. The default value is 16. Note, the value of Start Bit plus the value of Length must be less than or equal to 16.

Examples: The 16 contacts on a Relay Output card are designated contact no. 0 (LSB) through contact no.15 (MSB). Here are three examples of entering Start Bit and Length parameters to associate the desired contacts with a name. Only the Named contacts will be programmed by Output, Preset, Init, and System_init statements (Programming examples are given later).

1. To associate all 16 contacts with a name, Start Bit = 0 Length = 16
2. To associate contacts 3 through 5 with a name, Start Bit = 3 Length = 3.
3. To associate contact 10 with a name, Start Bit = 10 Length = 1.

Timeout

Specifies the time in seconds that will be allowed for the Relay Output card to complete its operation. If an Output or Wait_for statement does not complete within this time, a Timeout error message (error code 803) will be generated. The legal range for Timeout is from 0.001 and 128,000 seconds. The default value is 5 seconds. The Relay Output card completes operations in 6 milliseconds.

```
Name --- Relay                    Date: 21 Aug 1984
Model 69730A Relay Actuator       Time: 17:28:20
File Name -- Chapter_8
User ID ----
Time-stamp - First Time

                                  Page 2 of 2

Standard Data Type ........ Unsigned Binary
Data Conversion ........... Standard


Run Time Limit Checking ... No


Initial Value ............. 0
```

Menu Page 2 of 2

Standard Data Type

Specifies either "Unsigned Binary" or "2's Complement Binary" as the Standard Data Type for the Relay Output card. The default selection is Unsigned Binary.

1. Unsigned Binary. The user programs in positive numbers only. The 14752A Output and Preset statements interpret these numbers to be unsigned binary values for the Relay Output card. The range of numbers that can be programmed is from 0 to $2^N - 1$; where N is the number of "Named" contacts specified by the Start Bit and Length parameters.

Examples:

a. If all 16 contacts, 0(LSB) to 15(MSB), are associated with a name, the range is from 0 to $2^{16} - 1$, or from 0 (all 16 contacts opened) to 65535 (all 16 contacts closed).

b. If any 4 consecutive contacts(e.g. contacts 4 to 7) are associated with a name, the range is from 0 to $2^4 - 1$, or from 0 (all 4 contacts opened) to 15 (all 4 contacts closed).

NOTE

Although the 2's complement data type is available, the Relay Output card is normally programmed using the unsigned binary data type.

Menu Page 2 of 2 (continued)

2. 2's Complement Binary. The user can program in positive or negative numbers. The 14752A Output and Preset statements interpret the negative numbers as 2's complement binary values for the Relay Output card. The range of numbers that can be programmed is from $-2^{(N-1)}$ to $+[2^{N-1)}]-1$; were N is the number of "Named" contacts.

Examples:

a. If all 16 contacts, 0(LSB) to 15(MSB), are associated with a name, the range is from -2^{15} to $+2^{15}-1$, or from -32768 (all contacts open except contact 15) to 32767 (all contacts closed except contact 15).

b. If any 4 consecutive contacts (e.g. 4 to 7) are associated with a name, the range is from -2^3 to $+2^3-1$, or from -8 (all contacts opened except contact 7) to 7 (all 4 contacts closed except contact 7).

```
Name --- Relay                        Date: 21 Aug 1984
Model 69730A Relay Actuator           Time: 17:28:20
File Name -- Chapter_8
User ID ----
Time-stamp - First Time

                                  Page 2 of 2

Standard Data Type ........ Unsigned Binary
Data Conversion ........... Standard


Run Time Limit Checking ... No


Initial Value ............. 0
```

Menu Page 2 of 2 (continued)

Data Conversion

Specifies how the data value sent to the Relay output card by an Output or Preset statement is to be scaled. One of two options can be selected: Standard or User.

1. "Standard" conversion for this card means that the data values in an Output or Preset statement are sent to the card as numbers that represent unsigned binary or 2's complement values for the card, whichever was selected as the standard data type. Standard (unsigned binary) is the default data conversion for this function.

NOTE

Although the User option is available on all functions, it is not likely to be used when programming the Relay Output function. Refer to Chapter 3 for details on User Units.

2. "User" conversion allows the card data to be specified in terms of the application rather than in terms of the card's standard units. This means that you can program in user defined units and have the data scaled when it is sent to the card. As shown below, four additional entries appear on the menu when User conversion is selected.

User's
P1 P2

Standard
P1 P2

Appear on menu only when "User" conversion is selected. These entries allow you to fill in the ratio of User and Standard values so that the 14752A can convert the user defined data values into the equivalent Standard values which are recognized by the card. The P1 default values are 0; the P2 default values are 1. The Digital Output function described later in this Chapter includes an example that shows how to enter these values.

Run Time Limit Checking

If toggled on (Yes), the value sent to the Relay Output card will be checked against the High and Low Limits that you specify. The default for this parameter is off (No).

High Limit

Appears only when Run Time Limit Checking is on (Yes). Specifies the upper bound for the contact closure value. The value entered must be greater than the Low Limit entered. Error message 802 will be generated when a programmed value is greater than the High Limit entered. This value **MUST** be entered in the same units as selected under Data Conversion. This parameter **MUST** be filled in; it has no default value.

Low Limit

Appears only when Run Time Limit Checking is on (Yes). Specifies the lower bound for the contact closure value. The value entered lower than the High Limit entered. Error message 802 will be generated when a programmed value is lower that the Low Limit entered. This value **MUST** be entered in the same units as selected under Data Conversion. This parameter **MUST** be filled in; it has no default value.

Initial Value

Specifies the value which will be sent to the Relay Output card when it is initialized by an Init or System_init statement. This value **MUST** be entered in the same units as selected under Data Conversion. The default value is 0 Standard Units (all contacts opened).

Programming Fundamentals

Only two 14752A statements are required to program the Relay Output function. Each statement is described below and then a simple example program is given. For details concerning the syntax requirements for these subprograms, refer to Chapter 3.

Statement	Description
Init (Name)	Sets the contacts (Selected in menu by Start and Length) on the named Relay Output card to the Initial Value specified in the menu. Init also clears the EOP (End-of-Process) signal at the card's edge connector, disables interrupts, and resets the value of timeout to the value specified in the menu.
Output(Name, Value)	Sets the contacts (selected in menu by Start and Length) on the named Relay card to the Value specified in the Output Statement.

The sixteen relays on the 69730A card, designated Contact No. 0(LSB) through 15 (MSB), can be used as individual actuators for process control equipment, annunciators, and alarm circuits. The isolation between relays afforded by the card and the ability of the 14752A to program individual relays on a single card allow the control of up to 16 independent functions. For example, one relay may control a valve for a process control system while another relay may activate a panel light which indicates the opening and closing of the valve.

The Start and Length menu parameters allow individual contacts or groups of contacts on a single relay card to be programmed. A separate Name identifier is specified for each group (or single contact) that has different Start and Length parameters. The following Table shows an example of using the Name identifiers with different Start and Length parameters. This configuration is used in the program example on the next page.

Contact Nos.	Name	Start	Length
4 through 7	Valves	4	4
4 and 5	Valve_1	4	2
6 and 7	Valve_2	6	2

In this example, Valve__1 is controlled by Relay No.4 and its indicator light by Relay No. 5; Valve__2 is controlled by Relay No. 6 and its indicator light by Relay No. 7. All four relays are assigned the name "Valves" so that they can be initialized simultaneously with the Init statement. Note that the System__init statement could also be used to initialize all of the named functions. Refer to Chapter 3 for details on System__init as well as the COM (line 10) and the Configure (line 20) statements used in the example program below.

Programming Example for Controlling Valves

```
10 COM/Names/Valves,Valve__1,Valve__2
20 Configure("Ask__me", "FILENAME")
30 !
40 ! Initialize the valves using the default for the Initial
50 ! Value parameter (0, all open) which was specified in
60 ! the menus.
70 !
80 Init(Valves)
90 !
100! Set up variables to open and close the valves
110 Open = 0
120 Close = 3
130 !
140 ! Close Valve No.1, Indicator for Valve No. 1 on
150 !
160 Output(Valve__1,Close)
170 !
180 ! Close Valve No.2, Indicator for Valve No. 2 on
190 !
200 Output(Valve__2,Close)
210 !
220 ! Open both valves, both indicators off
230 !
240 Init(Valves)
250 ! Close Valve No.1, Indicator for Valve No. 1 on
260 !
270 Output(Valve__1,Close)
280 !
290 ! Open Valve No. 1, Indicator for Valve No. 1 off
300 Output(Valve__1,Open)
310 END
```

Advanced Programming

There are 14752A statements which allow special purpose programming of the Relay Output function. They are considered to be advanced programming statements because you require a good understanding of the Multiprogrammer System to put these statements together to form a meaningful program. In general, these statements are used only when precise control of timing is a concern. They allow features like parallel output of more than one function, interrupts, and external triggering(see Chapter 10 for details). The advanced programming statements that can be used to program the Relay Output card are described below. An example, using Preset and Start statements to simultaneously change the contact closures on several Relay Output functions, is given immediately following the statement descriptions.

Statement	Description
Preset(Name,Value)	Sends a data value to the named Relay card but does not cause the contacts to change. The Preset statement is used as a setup operation for the Start statement or for an External Trigger.
Start(Name1,...,Name8)	Sets the contacts on the named Relay Output functions to the pattern specified by the last value sent to the function. Contacts on all named Relay Output functions will be set at the same time.
Set_timeout(Name,Value)	Changes the value of Timeout from the time specified in the menu to Value. Time is specified in seconds, from 0.001 to 128,000.
Wait_for(Name)	Suspends program execution until the Relay Output card has been started and completed operations or until a timeout has occurred. The Timeout parameter in the menu specifies the Timeout period. A Timeout error (code 803) is generated if Timeout occurs before the card completes its operation. Used if you want to wait for the card to finish when it has been started with a Start statement or an External Trigger signal.
Check_done(Name,Flag)	Inquires about the state of the Relay Output card's operation. If the card has completed, the Flag returned is a "1". If the card has not completed, the value of Flag returned is a "0". Once a value of one is returned, the Check_done statement will subsequently return zero until the function is started again.
Enable_intr(Name, Vector)	Allows the 69730A Relay Output card to signal the computer via the interrupt system when it has finished operations and the relay contacts have settled. This statement must be used in conjunction with the interrupt statements as described in Chapter 10. The Vector variable identifies the function that caused the interrupt.
Disable_intr(Name)	Inhibits the Relay Output card from signaling the computer when it has completed operations. It is meaningful only when an Enable_intr statement has already been sent. Note that the Init and System_init statements will also cancel a previously issued Enable_intr statement.

Advanced Programming Example

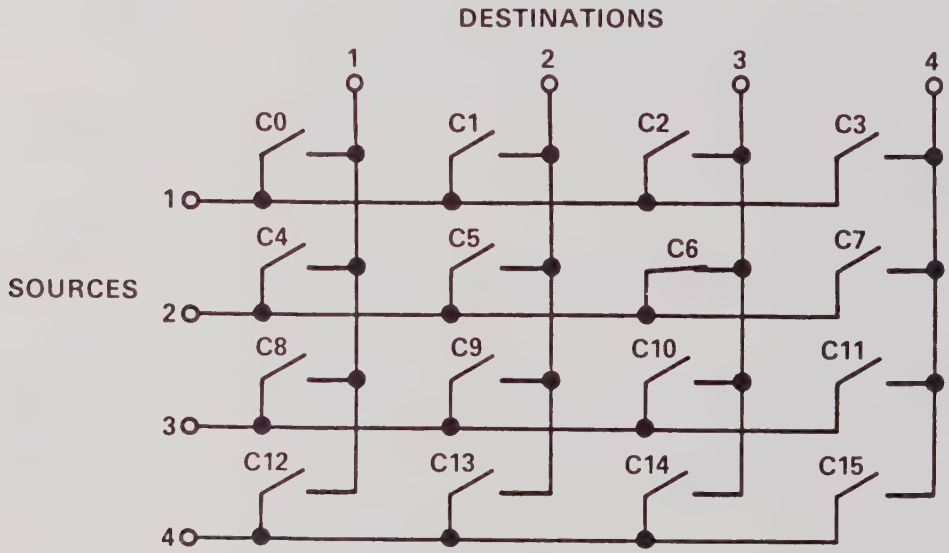
In the following example, three different Relay Output cards are assigned the names: Relay__1, Relay__2, and Relay__3. All 16 contacts (Start = 0, Length = 16) on each card are associated with the applicable name. First, a System__init statement is used to initialize all three cards to 0 (all contacts opened). Next, three Preset statements are used to send new contact closure data to the three cards. Then, a Start statement causes the relay contacts on each card to switch simultaneously to their new values.

```
10  COM/Names/Relay__1,Relay__2,Relay__3
20  Configure("Ask__me","MYFILE")
30  !
40  ! Initialize relays to Initial Value (default, all relays open)
50  ! in menus
60  System__init
70  !
80  ! Send the desired relay contact closure data to each card
90  ! Values are in Standard Units, Unsigned Binary Data Type.
100 !
110 Preset(Relay__1,4095)
120 Preset(Relay__2,2047)
130 Preset(Relay__3,1023)
140 !
150 ! Simultaneously cause the three functions to change their
160 ! Relay contact closures
170 !
180 Start(Relay__1,Relay__2,Relay__3)
190 !
200 END
```

Using the 69730A Relay Output Card as a Matrix Switch

In CAT Systems it is often necessary to connect various instruments such as a digital voltmeter to numerous test points within the system. The 69730A card can provide this feature. One or more Relay Output cards can be connected and configured to provide a programmable matrix for a computer aided test system.

The Relay card, as it functionally relates to matrix applications, is shown in the figure below. Although this configuration shows a 4X4 matrix, there are a large number of configurations possible, depending upon how many cards are used. The external wiring required for the implementation of a 4X4 matrix is as shown in the figure. A matrix can be programmed to connect various sources (e.g., voltmeters and signal generators) to destinations (or loads) which are the test points of the system. For example, closing contact C6 will connect source no. 2 to destination no. 3.



C0 (LSB) THROUGH C15 (MSB) ARE CONTACTS ON ONE 69730A CARD

4x4 Matrix Switch

Matrix Switch Programming Example

In the programming example that follows, the four sources have been identified with the Names: Source__1,Source__2,Source__3, and Source__4. Also, the contacts on the 69730A card have been associated with these Names using the Start and Length parameters as follows:

Contact Nos.	Name	Start	Length
0 to 3	Source__1	0	4
4 to 7	Source__2	4	4
8 to 11	Source__3	8	4
12 to 15	Source__4	12	4

The following program example is used to ouput data to our 4x4 relay matrix. The data values are set up so that a source can only be connected to one destination at a time. Note that there are many ways in which to program a relay switching matrix, the following is a simple example.

```
10    COM/Names/Source__1,Source__2,Source__3,Source__4
20    Configure("Ask__me","FILENAME__4")
30    System__init
40    ! Set up data values(Values are Standard units,unsigned binary)
50    !
60    Dest__1=1 ! Selects destination no. 1.
70    Dest__2=2 ! Selects destination no. 2.
80    Dest__3=4 ! Selects destination no. 3.
90    Dest__4=8 ! Selects destination no. 4.
100   !
110   Output(Source__1,Dest__4)          ! Closes contact C3 connecting
120                                      ! source no. 1 to destination no. 4.
130   !
140   Output(Source__2,Dest__1)          ! Closes contact C4 connecting
150                                      ! source no. 2 to destination no. 1.
160   !
170   Preset(Source__3,Dest__2)          ! Presets named contact group with
180                                      ! a new data value.
190   Preset(Source__4,Dest__3)          ! Presets named contact group with
200                                      ! a new data value.
210  Start(Source__3,Source__4)          ! Simultaneously closes contacts C9
220                                      ! and C14 connecting source no. 3
230                                      ! to destination no. 2; and connect
240                                      ! source no. 4 to destination no. 3
250   END
```

69731B Digital Output Card

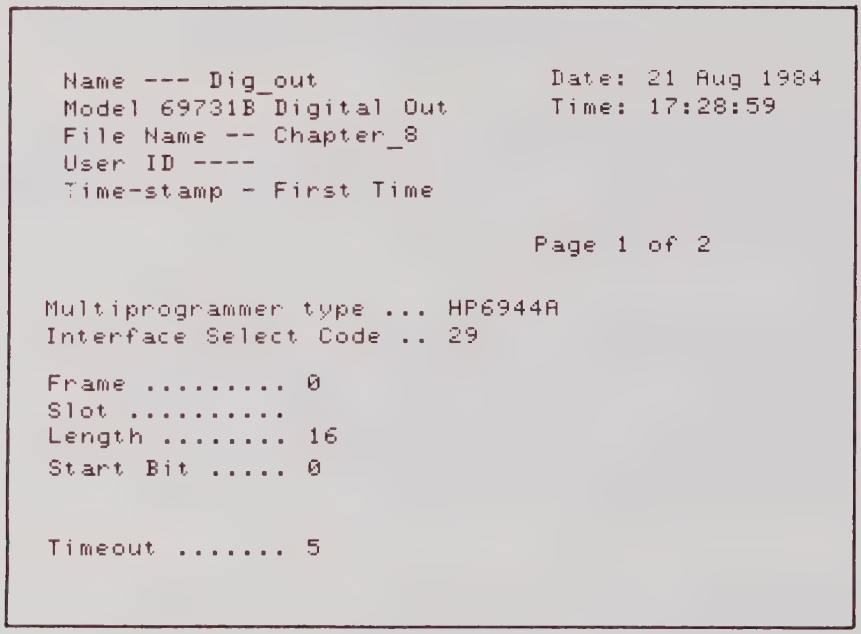
This card provides sixteen programmable output data lines which can be configured for: TTL, DTL, CMOS, or open collector currents. As shipped from the factory, the card is configured so that all 16 outputs are TTL positive true. A single jumper on the card can be used to invert the logic sense of all 16 output lines. Other jumpers on the card allow the outputs to be changed to one of the other available logic types (e.g., CMOS).

Gate and Flag handshake signals are available at the 69731B card's edge connector to synchronize data transfers between the Digital Output card and an external device. The 69731B card will not indicate that it has completed operations until it receives the Flag signal from the external device. As shipped from the factory, the 69731B card's Gate and Flag signals are jumpered together causing the function to complete instantaneously. The Gate and Flag signals as well as other edge connector signals are described in Chapter 10 of this manual. Refer to the 69731B Digital Output card Operating Manual, (HP Part No. 69731-90003) for complete hardware details including descriptions of jumper positions.

The following paragraphs describe the menu entries and 14752A statements that are used to program the 69731B Digtial Output card. In addition, a programming example using the 69731B Digital Output card to program the HP 14570A AC Power Controller is provided. The 14570A, equipped with Option 042, includes a modified 69731B card and an interconnecting cable.

Menu Entries for Model 69731B

The configuration and initial conditions for the 69731B Digital Output function are specified on a two page menu. The user is required to fill in the various parameters on the menu in order to program the function. Instructions on how to use the menus and additional information on most of the parameters are given in Chapter 3. The Digital Output function's parameters are described as follows.



Menu Page 1 of 2

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ..... 8
HP-IB Address ......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the Digital Output card. The legal range is from 0 to 7. The default value is 0.

Menu Page 1 of 2 (continued)

Slot

Identifies the location of the Digital Output card within the frame. The legal range is from 0 to 15. This parameter **MUST** be filled in; it has no default value.

Start Bit

Specifies the first Bit within a group of bits on the Digital Output card. The legal range is from 0 to 15. The default value is 0.

Length

Specifies the total number of bits, beginning with the Start Bit, in the group. The legal range is from 1 to 16. The default value is 16. Note, the value of Start Bit plus the value of length must be less than, or equal to, 16.

Examples: The 16 bits on a Digital Output card are designated Bit 0 (LSB) through Bit 15 (MSB). Here are three examples of entering Start Bit and Length parameters to associate the desired bit numbers with a name. Only the "named" bits will be programmed by Output, Preset, Init, and System_init statements (Programming examples are given later).

1. To associate all 16 bits with a name, Start Bit = 0 Length = 16.
2. To associate bits 3 through 5 with a name, Start Bit = 3 Length = 3.
3. To associate bit 10 with a named, Start Bit = 10 Length = 1.

Timeout

Specifies the time in seconds that will be allowed for the Digital Output card to complete its operation. If an Output or Wait_for statement does not complete within this time, a Timeout message (error code 803) will be generated. The legal range for Timeout is from 0.001 and 128,000 seconds. The default value is 5 seconds.

For the case in which the Digital Output card completes operations instantaneously (Gate/Flag signals are jumpered), the Timeout parameter is not meaningful for Output statements. However, if the jumper is removed and the Gate/Flag signals are used to synchronize the data transfer with an external device, the Timeout parameter could be meaningful. For example, assume that the external device requires 10 milliseconds to convert the digital data it receives from the 69731B card. For this example, do not enter a Timeout value that is less the 10 milliseconds (a .01 or 1E-2 entry in menu).

```
Name --- Dig_out                    Date: 23 Aug 1984
Model 69731B Digital Out            Time: 17:29:11
File Name -- Chapter_8
User ID ----
Time-stamp - First Time

                                 Page 2 of 2

Standard Data Type ........ Unsigned Binary
Data Conversion ........... Standard


Run Time Limit Checking ... No


Initial Value ............. 0
```

Menu Page 2 of 2

Standard Data Type

Specifies either "Unsigned Binary" or "2's Complement Binary" as the Standard Data Type for the Digital Output card. The default selection is Unsigned Binary.

1. Unsigned Binary. The user programs in positive numbers only. The 14752A Output and Preset statements interpret these numbers to be unsigned binary values for the Digital Output card. The range of numbers that can be programmed is from 0 to $2^N - 1$; where N is the number of "Named" Bits specified by the Start Bit and Length parameters.

Examples:

a. If all 16 bits, 0(LSB) to 15(MSB), are associated with a name, the range is from 0 to $2^{16}-1$, or from 0 (all 16 bits off) to 65535 (all 16 bits on).

b. If any 4 consecutive contacts(e.g. bits 4 to 7) are associated with a name, the range is from 0 to $2^{4}-1$, or from 0 (all 4 bits off) to 15 (all 4 bits on).

NOTE

Although the 2's complement data type is available, the Digital Output card is normally programmed using the unsigned binary data type.

2. 2's Complement Binary. The user can program in positive or negative numbers. The 14752A Output and Preset statements interpret the negative numbers as 2's complement binary values for the Digital Output card. The range of numbers that can be programmed is from $-2^{(N-1)}$ to $+[2^{N-1)}]-1$; were N is the number of "Named" bits.

Examples:

a. If all 16 bits, 0(LSB) to 15(MSB), are associated with a name, the range is from -2^{15} to $+2^{15}-1$, or from -32768 (all bits off except bit 15) to 32767 (all bits on except bit 15).

b. If any 4 consecutive bits (e.g. 4 to 7) are associated with a name, the range is from -2^{3} to $+2^{3}-1$, or from -8 (all bits off except bit 7) to 7 (all bits on except bit 7).

```
Name --- Dig_out                    Date: 23 Aug 1984
Model 69731B Digital Out            Time: 17:29:16
File Name -- Chapter_8
User ID ----
Time-stamp - First Time

                                  Page 2 of 2

Standard Data Type ........ Unsigned Binary
Data Conversion ........... Standard


Run Time Limit Checking ... No


Initial Value ............. 0
```

Menu Page 2 of 2 (continued)

Data Conversion

Specifies how the data value sent to the Digital Output card by an Output or Preset statement is to be scaled. One of two options can be selected: Standard or User.

1. "Standard" conversion for this card means that the data values in an Output or Preset statement are sent to the card as numbers in the form of unsigned binary or 2's complement values, whichever was selected as the standard data type. Standard (unsigned binary) is the default data conversion for this function.

2. "User" conversion allows the card data to be specified in terms of the application rather than in terms of the card's standard units. This means that you can program in user defined units and have the data scaled when it is sent to the card. As shown below, four additional entries appear on the menu when User conversion is selected.

User's
P1 P2

Standard
P1 P2

Appear on menu only when "User" conversion is selected. These entries allow you to fill in the ratio of User and Standard values so that the 14752A can convert the user defined data values into the equivalent standard values which are recognized by the card. The P1 default values are 0; P2 default values are 1.

Menu Page 2 of 2 (continued)

Example: Assume that the 16-bit Digital Output card is used to send data to an external 12-bit Digital to Analog Converter device that has an output from 0 to 20.475V and a resolution of .005v: .005 times 4095 (all 12 bits on) equals 20.475. In order to program the Digital Ouput function in volts instead of the Standard numbers,first select the Unsigned Binary Data Type and Start = 0 Length = 12, then select User data Conversions and make the following entries for P1 and P2.

	P1		P2
User's:	0	:	20.475
Standard:	0	:	4095

Run Time Limit Checking

If toggled on (Yes), the value sent to the Digital Output card will be checked against the High and Low Limits that you specify. The default for this parameter is off (No).

High Limit

Appears only when Run Time Limit Checking is on (Yes). Specifies the upper bound for the output data values.The number entered must be greater than the Low Limit entered. Error message 802 will be generated when a programmed value is greater than the High limit entered. This value **MUST** be entered in the same units as selected under Data Conversion. This parameter **MUST** be entered; it has no default value.

Low Limit

Appears only when Run Time Limit Checking is on (Yes). Specifies the lower bound for the output data values. The value entered must be lower than the High Limit entered. Error message 802 will be generated when a programmed value is lower that the Low Limit entered. This value **MUST** be entered in the same units as selected under Data Conversion. This parameter **MUST** be entered; it has no default value.

Initial Value

Specifies the value which will be sent to the Relay Output card when it is initialized by an Init or System_init statement. This value **MUST** be entered in the same units as selected under Data Conversion. The default value is 0 Standard units (all bits off).

Programming Fundamentals

Only two 14752A statements are required to program the Digital Output function. Each statement is described below and then two simple example programs are given. For details concerning the syntax requirements for these statements, refer to Chapter 3.

Statement	Description
Init(Name)	Sets the output bits (Selected in menu by Start and Length) on the named Digital Output card to the Initial Value specified in the menu. Init also clears the EOP (End-of-Process) signal at the card's edge connector, disables interrupts, and resets Timeout to the value specified in the menu.
Output(Name, Value)	Sets the output bits (Selected in menu by Start and Length) on the named Digital Output card to the Value specified in the Output Statement.

The sixteen output bits on the 69731B card, designated B0 (LSB) through B15(MSB), can be programmed on or off individually or in a group that consists of two or more (up to 16 bits). The Start and Length menu parameters allow individual bits or groups of bits on a single card to be programmed. A separate Name identifier is specified for each group(or single bit) that has different Start and Length parameters. The following examples show how to program a Digital Output card using Name identifiers with different Start and Length parameters.

Example 1: In this example, all 16 bits (Start = 0, Length = 16) have been assigned the name "Dig_out" and the default values on the menu are in effect. The following program first sets up the variables to hold the Odd_bit and Even_bit data values and initializes the function (all bits off, logic zeros). Next, an Output statement is used to turn on the odd numbered bits. Then, another Output statement is used to turn the odd numbered bits off and the even numbered bits on. Refer to Chapter 3 for details on the COM and Configure statements shown in program lines 10 and 20 respectively.

```
10 COM/Names/Dig_out
20 Configure("Ask_me", "FILENAME")
30 !
40 ! Set up variables to turn on the odd bits and then turn on
50 ! the even bits. Odd_bits are bits 1,3,5,7,9,11,13,15 and
60 ! Even_bits 0,2,4,6,8,10,12,14
70 !
80 Odd_bits=43690      ! Values are in Standard units, unsigned binary
90 Even_bits=21845
100 ! Initialize the output data bits using the default (0, all
110 ! bits off) for the Initial Value parameter in the menu.
120 !
130 Init(Dig_out)
140 !
150 ! Turn on the odd numbered bits.
160 !
170 Output(Dig_out, Odd_bits)
180 !
190 ! Turn off the odd numbered bits and turn on the
200 ! even numbered bits.
210 !
220 Output(Dig_out,Even_bits)
230 !
240 END
```

Example 2: In this example, the 16 bits on a single Digital Output card have been assigned names as follows:

Bit Nos.	Name	Start	Length
0 through 3	Dig_A	0	4
4 through 7	Dig_B	4	4
8 through 11	Dig_C	8	4
12	Dig_12	12	1
13	Dig_13	13	1
14	Dig_14	14	1
15	Dig_15	15	1

The configuration shown above is used in the following program example. The program first sets up the variables for each Name and initializes the function (all 16 bits off). Next, Output statements are used to send the appropriate data to each Name. Each Output statement only affects the named bit(s).

```
10  COM/Names/Dig_A,Dig_B,Dig_C,Dig_12,Dig_13,Dig_14,Dig_15
20  Configure ("Ask_me","Filename_2")
30  !
40  ! Set up data variables. Values are in Standard units unsigned binary.
50  Data_A=9 ! Data value for group A
60  Data_B=6 ! Data value for group B
70  Data_C=7 ! Data value for group C
80  On=1 ! Data value to turn on an individual bit
90  Off=0 ! Data value to turn off an individual bit
100 !
110 ! Initialize bits to Initial Value (default, all bits off)
120 !
130 System_init
140 !
150 ! Send data values to the desired groups
160 !
170 Output(Dig_A,Data_A) ! Turns on bits 0 and 3; turns off bits 1 and 2
180 Output(Dig_B,Data_B) ! Turns on bits 5 and 6; turns off bits 4 and 7
190 Output(Dig_C,Data_C) ! Turns on bits 8,9,and 10; turns off bit 11
200 !
210 ! Turn individual bits on and off
220 !
230 Output(Dig_12,On)      ! Turn bit 12 on
240 Output(Dig_13,On)      ! Turn bit 13 on
250 Output(Dig_12,Off)     ! Turn bit 12 off
260 Output(Dig_14,On)      ! Turn bit 14 on
270 Output(Dig_15,On)      ! Turn bit 15 on
280 !
290 END
```

Advanced Programming

There are 14752A statements which allow special purpose programming of the Digital Output function. They are considered to be advanced programming statements because a good understanding of the Multiprogrammer System is required to put these statements together to form a meaningful program. In general, these statements are used only when precise control of timing is a concern. They allow features like parallel output of more than one function, interrupts, and external triggering (see Chapter 10 for details). The advanced programming statements that can be used to program the Digital Output card are described below. An example, using Preset and Start statements to simultaneously change the output bits on several Digital Output cards, is given immediately following the statement descriptions.

Statement	Description
Translate(Name, Iarray(*),User(*))	Used in conjunction with Output_iblock to process data for the Digital Output card according to the specified data conversion. Data is taken from the REAL array named "User", converted to Integer numbers, and returned into the INTEGER array named "Iarray". All arrays **MUST** have been previously DIMensioned or ALLOCATEd in the user's program otherwise a BASIC system error (#13) will be generated.
Output_iblock(Name, Iarray(*)[,Count[,Index]])	Used to send integer data from the array named "Iarray", directly to the Digital Output card. Since this array is the same array as "Iarray" used in the Translate statement, the array names are the same. The number of words sent to the card is equal to the size of the data array. The optional parameter Count, specifies the number of words that will be sent to the card. This number must be less than or equal to the size of the array. Along with Count, Index can be used to specify the array location of the first data item that will be sent. The data array **MUST** have been previously DIMensioned or ALLOCATEd in the user's program otherwise a BASIC system error (#13) will be generated. This statement presets and starts the card and waits for a Gate/Flag handshake to complete (see Chapter 10 for Gate/Flag handshake details).
Preset(Name,Value)	Sends a data value to the named Digital Output card but does not cause the output bits to change value. The Preset statement is used as a setup operation for the Start statement or for an External Trigger.
Start(Name1,...,Name8)	Sets the output bits on the named Digital Output cards to the pattern specified by the last value sent to the card. Output bits on all named Digital Output cards will be changed at the same time.
Set_timeout(Name, Value)	Changes the value of Timeout from the time specified in the menu to Value. Time is specified in seconds, from 0.001 and 128,000.

Wait_for(Name)

Suspends program execution until the Digital Output card has been started and completes operations or until a timeout has occurred. The Timeout parameter in the menu specifies the timeout period. A Timeout error #803 is generated if Timeout occurs before the card completes its operation. Used if you want to wait for the card to finish when it has been started by a Start statement or an External Trigger signal.

Check_done(Name, Flag)

Inquires about the state of the Digital Output card's operation. If the card has completed, the value of Flag returned is a "1". If the card has not completed, the value returned is a "0". Once a value of one is returned, the Check_done statement will subsequently return zero until the function is started again.

Enable_intr(Name, Vector)

Allows the 69731B Digital Output card to signal the computer via the interrupt system when it has finished operations. This statement must be used in conjunction with the BASIC statement ON INTR as described in Chapter 10. The Vector variable identifies the function that caused the interrupt.

Disable_intr(Name)

Inhibits the Digital Output card from signaling the computer when it has completed operations. It is meaningful only when an Enable_intr statement has already been sent. Note that the Init and System_init statements will also cancel a previously issued Enable_intr statement.

Advanced Programming Example

In the following example, two different Digital Output cards are assigned the names: Digout_1 and Digout_2. All 16 bits (Start = 0, Length = 16) on each card are associated with the applicable name. First, a System_init statement is used to initialize both cards to 0 (all bits off). Next, two Preset statements are used to send new data values to the two cards. Then, a Start statement causes the output bits on each card to change simultaneously to their new values.

```
10  COM/Names/Digout_1, Digout_2
20  Configure ("Ask_me","MYFILE")
30  !
40  ! Initialize cards to Initial Value (default, all bits off)
50  !
60  System_init
70  !
80  ! Set up data values to turn on the odd numbered bits on one
90  ! card and the even numbered bits on the other card.
100 !
110 Odd_bits = 43690 ! Values are in Standard units, unsigned binary
120 Even_bits = 21845
130 !
140 ! Send the desired data values to each card
150 !
160 Preset(Digout_1,Odd_bits)
170 Preset(Digout_2,Even_bits)
180 !
190 ! Simultaneously change the output bits on the two cards.
200 ! Turns on the odd numbered bits (1,3,5, etc.) on the
210 ! card named "Digout_1" and turns on the even numbered
220 ! bits (0,2,4, etc.) on the card named "Digout_2".
230 !
240 Start(Digout_1,Digout_2)
250 !
260 END
```

Using the 69731B to Program the 14570A AC Power Controller

The 14570A AC Power Controller is a digitally-controlled ac line switching and distribution unit for 12 ac outputs. The 12 outputs are partitioned into three groups of four outputs each. An individual output within a group can supply up to 6 amps but the total load current for any group cannot exceed 15 amps. Each output is individually controlled by a TTL signal from the 69731B Digital Output card. The outputs within each group, that is the four ac outputs, can also be controlled by a single digital input signal. Refer to the 14570A Operating and Service Manual (HP Part No. 14570-90001) for hardware details.

The 14570A with Option 042 includes the AC Power Controller, a 69731B Digital Output card, and a cable to interconnect the Power Controller with the 69731B card. This simplifies connecting the 69731B card's output data bits to the Power Controller's input control lines. The 69731B card provided with Option 042 has been modified for negative-true (LO true) logic. The standard 69731B card can easily be modified to provide negative-true output bits (Refer to the 69731B Operating Manual, HP Part No. 69731-90003).

If you are fabricating a cable to interconnect the 69731B and the 14570A, the following connections are required. Keep in mind that the inputs to the 14570A must be negative true.

Connections Between the 69731B and 14570A

69731B Output Connector		**14570A Input Connector**	
Pin	Signal	Pin	Signal
A	B0	1	CTRL0 LINE C
B	B1	2	CTRL1
C	B2	3	CTRL2
D	B3	4	CTRL3
E	B4	5	CTRL4 LINE B
F	B5	6	CTRL5
H	B6	7	CTRL6
J	B7	8	CTRL7
K	B8	9	CTRL8 LINE A
L	B9	10	CTRL9
M	B10	11	CTRL10
N	B11	12	CTRL11
P	B12	16	SYSDIS C
R	B13	20	SYSDIS B
S	B14	24	SYSDIS A
r	COM	14	Common 1
r	COM	18	Common 2
r	COM	22	Common 3

The first task to be performed when using the 14570A AC Power Controller with the 69731B Digital Output card is to connect loads to the outputs of the Power Controller. The connections should be made on the basis of equalizing the load currents so that the maximum ratings are not exceeded. The connection procedure is simplified by use of the 14570A Connection Record which can be found in the 14570A Operating and Service Manual.

Considerations should also be given to the disabling of the 14570A ac outputs using the SYSDIS input lines (A,B,C) on the 14570A. Those outputs which are to be disabled at the same time must be in the same group (A, B, or C).

The following program example uses a 69731B/14570A combination to control the application of ac power to a hypothetical process. In this example, two motor driven Fill Valves (A and B), four Vent Fans (A,B,C, and D), and two Drain Pumps(A and B) are controlled by the 69731B/14570A. The 14570A ac output assignments for this hypothetical process are listed in the following table along with the associated 69731B output bit and 14570A control line.

69731B Output Bit No.	**14570A Control Line**	**14570A AC Output**	**Description**
B0	CTRL0	Line C, Output 0	Fill Valve A (2 amps)
B1	CTRL1	Line C, Output 1	Fill Valve B (2 amps)
B5	CTRL5	Line B, Output 5	Vent Fan A (1 amp)
B6	CTRL6	Line B, Output 6	Vent Fan B (1 amp)
B7	CTRL7	Line B, Output 7	Vent Fan C (1 amp)
B9	CTRL9	Line A, Output 9	Drain Pump A (3 amps)
B10	CTRL10	Line A, Output 10	Drain Pump B (3 amps)
B11	CTRL11	Line A, Output 11	Vent Fan D (2 amps)
B12	SYSDIS C	N/A	Disables Line C Outputs
B13	SYSDIS B	N/A	Disables Line B Outputs
B14	SYSDIS A	N/A	Disables Line A Outputs

The assignments shown above were made on the basis of load balancing and ease of disabling the outputs. For example, programming bit B12 on sets the SYSDIS C line and disables Outputs 0 and 1 without disturbing the other outputs.

In this example, bits on the 69731B card have been assigned appropriate names using the Start and Length menu parameters as follows:

Bit Nos.	Name	Start	Length
B0 to B1	Valves	0	2
B5 to B7	Vents	5	3
B9 to B13	Master__control	9	5
B14	Pump__control	14	1

Fill Valves A and B are identified by the name "Valves"; Vent Fans A Through C are all identified by the name "Vents"; "Master__control" identifies Drain Pump A, Drain Pump B, Vent Fan D, and SYSDIS control lines B and C; "Pump__control" identifies SYSDIS control line A. This configuration is used in the program example shown on the next page.

Programming Example (continued)

```
10  COM/Names/Valves,Vents,Master__control,Pump__control
20  Configure("Ask__me","FILENAME__3")
30  !
40  ! Set up data variables
50  !
60  Open = 3
70  Closed = 0
80  On = 7
90  Off = 1
100 Shut__down = 31
110 Delay = .035
120 !
130 System__init ! Initialize system using default menu values:
140 ! valves closed, vents turned-off, drain pumps turned off.
150 !
160 Output(Vents,On) !Turns on vent fans A,B,and C.
170 WAIT Delay !There is a delay between the time a turn on
180 ! control signal is received and the ac output is activated.
190 ! This delay has a range from 6 to 30 milliseconds. There is
200 ! also a delay from 14 to 34 milliseconds between a turn-off
210 ! control signal and the ac output being deactivated.
220 !
230 Output(Valves,Open) ! Opens Valves A and B.
240 !
250 WAIT 5 ! Leave valves opened for 5 seconds.
260 !
270 Output(Master__control,Shut__down) ! Close valves A and B;
280 ! turn off vents A,B, and C; turn on drain pumps A and B;
290 ! turn on vent fan D.
300 !
310 WAIT 5 ! Let the drain pumps run for 5 seconds.
320 !
330 Output(Pump__control,off) !Turn off drain pumps A and B
340 Output (Master__Control,24) ! Turn off ven fan D.
350 !
360 END
```

Chapter 9
Timebase, Interrupt, and Memory Functions

Introduction

The Multiprogrammer system offers many diversified functions via its general purpose I/O cards. Many of these cards have the same, or superior, capabilities as large programmable instruments such as function generators, pulse generators, and counters. Other functions, like digital buffering and data analysis, available from Multiprogrammer I/O cards, are traditionally found only in specialized digital instruments and computers. This chapter covers a collection of these functions that are implemented by six different I/O cards.

The sequencing and synchronization of events require the generation of precision timing reference signals. The Multiprogrammer provides timing functions by using either the 69735A Pulse Train output card or the 69736A Timer/Pacer card. The 69735A card generates a pulse train having a programmable number of pulses and a programmable pulse width. The 69736A Timer/Pacer card generates either a single output pulse of a programmable pulse width or a continuous pulse train having a programmable frequency. The 69750A Scanner Control card, covered in Chapter 5, also provides timing capabilities.

The Timer function is a special purpose timebase that incorporates a Model 69736A Timer card and one, or two, Model 69775A Counter cards. It generates precision timing signals with a broad range of programmable frequencies and pulse train durations. The Timer is particulary useful for pacing a Buffered A/D converter because the timing pulses can run continuously. This allows continuous data acquisition and, upon receipt of an external signal, either stops the timing pulses immediately or after a programmable delay. Therefore data can be stored both before and after an event.

A counter can be used to count occurrences of a particular signal in a electronic circuit, count events in a production process, or perform frequency measurements. The counting function is supported by the model 69775A Counter/Totalizer card. This card has analog comparators at its inputs, allowing the selection of a particular threshold for its count and enable inputs. Using this capability, it can count contact closures, digital logic level pulses, or analog waveform transitions. Signals can be applied simultaneously to the card's count up and count down inputs at rates up to 1MHz. The enable inputs can be used to gate the counter on, or off, in a variety of modes.

The 69776A Interrupt card can be used for "on line" data analysis or event sensing. The card is essentially a digital comparator that can be programmed so that once a predetermined relationship occurs, an interrupt to the computer will be generated. The sixteen bit input word that caused the interrupt is stored on the card for readback to the computer. A mask register allows any of the bits to be eliminated from the comparison.

Introduction (continued)

The 14752A programming package provides statements that are tailored to the testing of only a single bit on the 69776A card. These statements are particularly useful when you want to determine the sense of the signal that caused the interrupt. The interrupting word can be read back to the computer to determine which direction the bit toggled. This process is called event sensing.

The ability to store and return digital data is the characteristic of a Memory Function. The data, in the form of a digital "word", is written into the memory function either by a controller such as a Series 200 computer or by an external device. An additional feature of the memory function is the ability to operate independently of the computer. The Multiprogrammer memory function is implemented either with the 69790B Memory card or the 69791A Memory Card System. The Memory card provides 4096 memory locations for data buffering and the Memory Card System can provide a maximum of 1,048,576 memory locations. The data word can be up to 16 bits wide. Various modes of operation are available that allow the memory to receive data from a computer or external device. The modes of the 69790B Memory card and the 69791A Memory Card System are explained later in this chapter.

The Buffer menu provides a means of using the Memory card or Memory Card System together with an input card or an output card. This combination of cards is knows as a buffered multiple card function. A complete list of the multiple card functions supported by the 14752A Programming Package can be found in Chapter 10. The Buffer menu will be discussed further in this chapter.

69735A Pulse Train Output Card

The 69735A Pulse Train Output card generates a programmable number of output pulses at a programmable frequency. Programming a positive number will produce pulses at the "positive" output port, programming a negative number will produce pulses at the "negative" output port. Pulse trains of up to 32,767 pulses can be produced. When the positive output port and the negative output port are connected to a stepper motor translator, the card controls the motor's direction, speed, and amount of rotation.

The output pulse train frequency is programmed in terms of the period of one cycle, which can range from 1 μsec. to 6.5535 seconds. The period is crystal controlled for a period accuracy of ±0.01% ± 100 nsec. Changing the period will change the speed of the stepper motor's rotation. The precision of the 69735A Pulse card output also makes it very suitable for use as a timebase generator. This function card is considered complete when the last programmed pulse has finished. More information (card output pins, etc.) on the model 69735A is contained in its Operating Manual (HP part number 69735-90003).

Menu Entries For Model 69735A

Before programming the 69735A Pulse Train output card, you must specify various parameters in the Configuration Menu for the card. Following are descriptions of all the parameters required to configure this function. Most of the parameters on the menus, as well as how to use the menu keys, are described in detail in Chapter 3.

```
Name --- Pulse_train              Date: 21 Aug 1984
Model 69735A Pulse Train          Time: 17:35:37
File Name -- Chapter_9
User ID ----
Time-stamp - First Time

                                  Page 1 of 3

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot ..........



Timeout ....... 5
```

Menu Page 1 of 3

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ...... 8
HP-IB Address .......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the Pulse Train output card. It has a legal range from 0 to 7. The default value is 0.

Slot

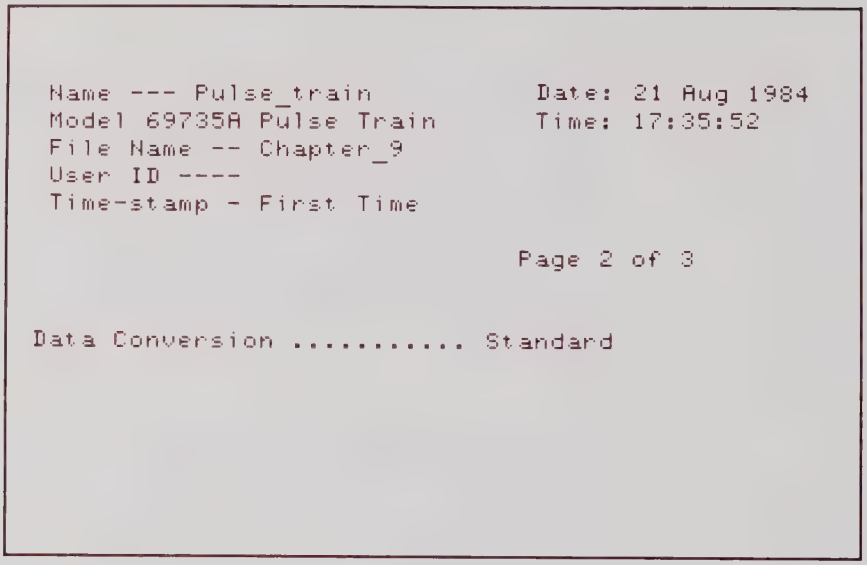
Identifies the location of the Pulse Train Output card within the frame. It has a legal range from 0 to 15. This parameter **MUST** be filled in; it has no default.

Menu 1 of 3 (continued)

Timeout

Sets the length of time (in seconds) that will be allowed for the Pulse Train Output output card to complete its operation. If the Output or Wait_for statements do not detect completion within this time, a Timeout error message (code #803) is generated. Any number between 0.001 and 128,000 may be entered here. The default value is only 5 seconds and must be changed if you wish to program pulse trains with durations greater than 5 seconds with an Output statement. The Wait_for statement is used for software synchronization of the hardware functions and is discussed in greater detail in Chapter 10.

Note that the maximum value of 128,000 does not cover the maximum time that is required for the card to complete an operation of generating 32767 pulses of 6.5535 second duration. This operation would require a total of 214,739 seconds. If the maximum range of the card is desired, then you must use the Preset and Start statements that are described in the Programming sections of this chapter.



Menu Page 2 of 3

Data Conversion

Specifies how the count value is to be scaled by the Output and Preset statements. Note, there is no User Conversion option available for Period. Period is always specified in seconds. One of two data conversion options can be selected: Standard and User.

1. "Standard" conversion is the default for this function. Standard units are in terms of "number of pulses". The range of allowed values is −32767 to 32767.

2. "User" allows data to be specified in terms of the application rather than in terms of the card's Standard Units. As shown below, four additional entries appear on the menu when User conversion is selected.

User's Units
P1 P2

Standard
P1 P2

P1 and P2 are only used when the User Conversion option is selected. These parameters allow you to fill in the ratio of User units and Standard units values so the 14752A can convert the user data into information that can be recognized by the card. The default values for P1 are 0; for P2 they are 1.

More About Data Conversion

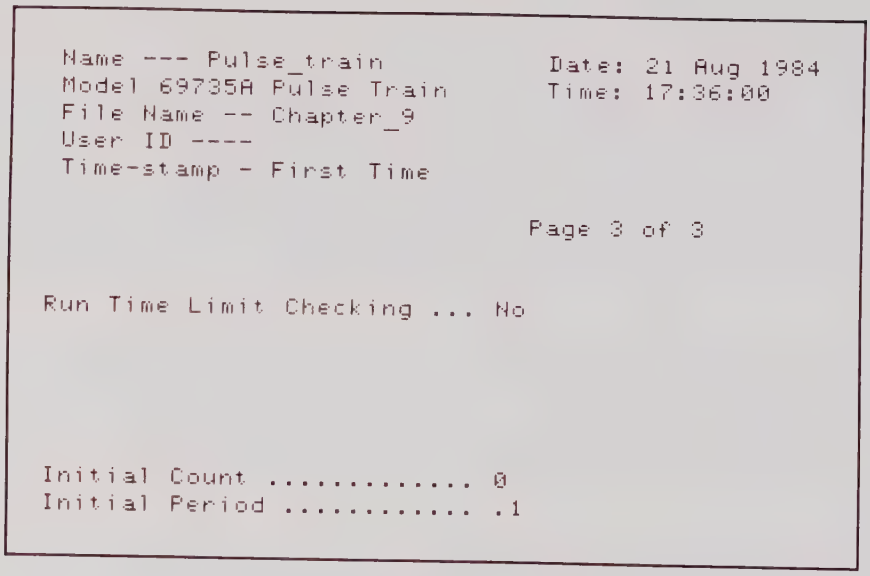
To further illustrate the use of data conversion, assume that the 69735A Pulse Train card is controlling a stepper motor through a stepper motor translator. We would like to program the function in degrees of rotation of the stepper motor (e.g. 0 to 360), but the Pulse Train card expects a value representing a certain number of pulses. In addition, a stepper motor translator's characteristics is specified in terms of a certain number of pulses per revolution. By selecting User conversion in the configuration menu, the 14752A CAT Programming Package will take the value in degrees of rotation (User's Units) and transform the value into "number of pulses" (Standard Units) before sending the data to the card.

From this example it can be seen that Standard conversion is used when you want to program a function by using numbers it expects. In other words, Standard units are the fundamental units of a function, number of pulses in our case above. Standard conversion is also the fastest way to get data to this function. An Output or Preset will execute 20% to 30% faster with Standard conversion than with User conversion.

User conversion gives us a much easier way to get data to a function, but it takes the 14752A software 20% to 30% longer to process the data. When User's conversion is selected in the menu, four other parameters must also be specified. These parameters give the 14752A all the data it needs to perform the necessary conversion calculations. For, example, if the stepper motor translator specification is 400 pulses per revolution, then to make the motor go one revolution we must give it 400 pulses. This means the range of values would be as follows:

	zero position (P1)	one revolution (P2)
User's Units	0 degrees	360 degrees
Standard Units	0 pulses	400 pulses

From this table we can see that:
User's P1 = 0
User's P2 = 360
Standard P1 = 0
Standard P2 = 400



Menu Page 3 of 3

Run Time Limit Checking

If YES is selected, values sent to the Pulse Train card will be checked against the specified Maximum and Minimum Period and maximum/minimum Count limits. This optional parameter can be toggled on or off by selecting YES or NO. Default is NO limit checking.

Maximum Count

Is used only when Run Time Limit Checking is ON. Specifies the upper limit for the pulse count. The value entered in the menu must be greater than the minimum pulse count specified. An error message (#802) will be generated when a programmed value is greater than this limit. This value should be entered in the same units that are selected for programming units. This parameter **MUST** be filled in; it has no default.

Minimum Count

Is used only when Run Time Limit Checking is ON. Specifies the lower limit for the pulse count. An error message (#802) will be generated when a programmed value is less than this limit. The value entered in the menu must be less than the Maximum pulse count specified and in the same units that are selected for programming units. This parameter **MUST** be filled in; it has no default.

Maximum Period

Is used only when Run Time Limit Checking is ON. Specifies the upper limit for the period data sent to the card. An error message (#802) will be generated when a programmed data value exceeds this limit. The value entered in the menu must be greater than the minimum period specified. The default value for this parameter is the maximum programmable period of 6.5535 seconds.

Menu Page 3 of 3 (continued)

Minimum Period

Is used only when Run Time Limit Checking is ON. Specifies the lower limit for the pulse width data sent to the function. An error message (#802) will be generated when a programmed data value is less than this limit. The value entered in the menu must be less than the Maximum period specified. The default value is the minimum programmable period of one microsecond (1E-6 seconds).

Initial Count

Specifies a Value representing the number of pulses to be Preset to the Pulse Train card when the Init or System_init statement is executed. It has a legal range from −32767 to 32767. This value should be entered in the same units that are selected for programming units. The default value is 0.

Initial Period

Specifies the period (in seconds) of the output pulse(s) when Init or System_init statement is executed. It has a legal range from 1 microsecond to 6.5535 seconds. The default value is 100 milliseconds.

Programming Fundamentals

Only a few fundamental statements are needed to program the Pulse Train Output card. The following is a summary of these statements. For details concerning the syntax requirements for the Name and Value parameters, refer to Chapter 3.

Statement	Description
Init(Name)	Loads the card with the initial count and period values which were previously specified in the configuration menu. It also aborts any pulse train currently being generated, disables interrupts, and resets Timeout to the value specified in the menu.
Set_period(Name,Value)	Sets the output pulse period. The acceptable range for Value is 1μsec. to 6.5535 seconds. The smallest increment that can be programmed is one microsecond.
Output(Name,Value [,Period])	Sends a data value to the function and starts the pulse train output. Causes the Pulse Train Output card to generate the number of pulses specified by Value. This statement will finish executing when the last pulse is done. Note, programming a value of zero will not generate any pulses. Period is an optional parameter that specifies the period of the generated output pulse(s). The range of values for Period is the same as for the Set_period statement (described above).
Stop(Name)	Aborts the pulse train currently being generated. The period value is not affected. .

Example 1. Using the 69735A as a stepper motor controller.

The features designed into the 69735A Pulse Train Output card allow the card to control stepper motors when used with a translator. The stepper motor's direction, speed and the amount of rotation can be controlled by the Pulse Train card. The direction is controlled by programming either a positive or a negative number to the card. Programming the pulse period will set the rotational speed of the motor while the amount of rotation is determined by the number of output pulses. The following example illustrates how to program the 69735A Pulse output card to control a stepper motor.

The Pulse Train Output card is identified with the Name "Motor" in this example. During the Configuration phase where the Name is defined, the initial conditions are set. These conditions include the pulse period which sets the speed of the stepper motor. Any translation factors such as pulses per revolution can be set so that programmed data will be in User Units. If we use the programming units explained earlier, rotating the motor 45 degrees in the clockwise direction would involve sending a value of +45 to the Pulse train card.

Programming Example for Stepper Motor Control

```
10 COM/Names/Motor
20 Configure ("Ask_me","MYFILE")
30 !
40 ! First, let's initialize the stepper motor
50 ! using the default Initial Period (100 milliseconds)
60 ! which was specified in the menu.
70 !
80 Init(Motor)
90 !
100 ! Let's rotate the motor +45 degrees relative to its current
110 ! position by using the default period specified in the
120 ! menu (100 milliseconds).
130 Output(Motor,45)
140 !
150 ! Now let's change the speed by changing
160 ! the period to 200 milliseconds.
170 !
180 Set_period(Motor,.200)
190 !
200 ! Now to rotate the motor, at the new
210 ! speed, counterclockwise 30 degrees
220 ! relative to its current position
230 !
240 Output(Motor, -30)
250 !
260 ! The motor is now positioned pointing
270 ! +15 degrees from the original starting position.
280 END
```

Example 2. Using the 69735A as a timebase generator.

A timebase is a device which produces a signal having a very precise frequency or pulse period. The 69735A Pulse Train card can be used effectively as a timebase because of its crystal-controlled pulse period. The output pulses can be used to trigger an instrument at a known rate or to externally trigger another Multiprogrammer function card.

The Buffered A/D function, discussed in this chapter, is a good example of a function which can be externally triggered by a 69735A Pulse Train Output card. Briefly, the Buffered A/D takes a voltage reading and places it in a memory buffer every time it is triggered. The 69735A can supply the Buffered A/D function with that trigger (i.e., the timing signal that initiates the data conversion. The number of data conversions and the time interval between data conversions are the two key parameters tor the 69735A timebase. The number of data conversions is specified by the Output statement and the time interval is specified by the Set_period statement.

In the following programming example assume that the 69735A has been assigned the name Time_base, and Standard Units was selected in the configuration menu.

Programming Example for a 69735A Timebase

```
10 COM/Names/Time_base
20 Configure ("Menu_on","MYFILE")
30 !
40 ! First let's set up the period to 50
50 ! microseconds between voltage samples.
60 !
70 Set_period(Time_base,50E-6)
80 !
90 ! Next, lets send 4000 pulses
100 ! to take 4000 readings.
110 !
120 Output(Time_base,4000)
130 !
140 ! If we want to change the time interval to 100
150 ! microseconds and take 200 readings at this rate
160 !
170 Set_period(Time_base,100E-6)
180 Output(Time_base,200)
190 END
```

Advanced Programming

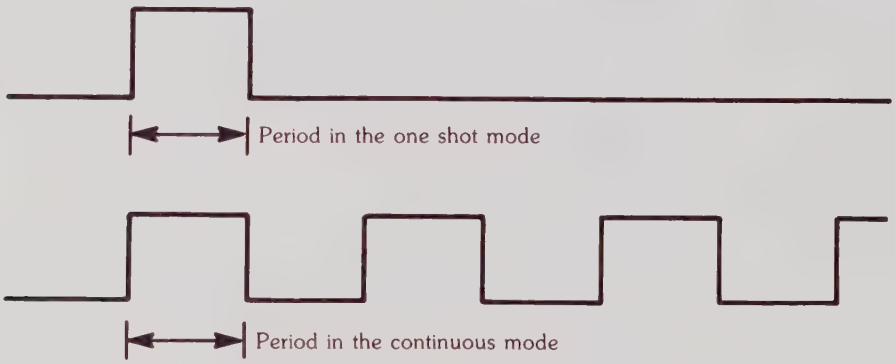
There are 14752A statements which allow special purpose programming of the Pulse Train Output function. They are considered advanced programming statements because you need a good understanding of the Multiprogrammer I/O cards and system to put together a meaningful program. In general, these statements are used only when precise control of the timing is a concern. They allow features such as: parallel outputs from more than one function card, external triggering, checking status, and interrupts. All of these features are explained in greater detail in Chapter 10.

Statement	Description
Preset(Name,Value [,Period])	Sends a pulse count value to the function but does not start the output pulse train. Used as a set up operation for a Start statement or an External Trigger. Period is an optional parameter that specifies the desired pulse period. The range of values for Period is the same as for the Set_period statement. Finishes quickly because it does not have to wait for a response from the card.
Start(Name1,..,Name8)	Starts the output pulse train with the number of pulses specified by the last Value sent to the card. If more than one Name is listed, all the named functions start in parallel and at the same instant. All of the Names must be located at the same Select Code. This statement does not wait for the pulse train to complete, therefore it finishes quickly. That is, with the Start command the program will continue even if the pulse train has not finished.
Wait_for(Name)	Program execution is paused until the function has completed its operation or a timeout occurs. Used if you want to wait for the Pulse Train Output card to finish when it has been started with a Start statement or an External Trigger.
Check_done(Name,Flag)	Inquires about the status of the function's operation. Sets Flag to one if the pulse train has completed. Once a value of one is returned, the Check_done statement will subsequently return zero until the function is Started again.
Enable_intr(Name,Vector)	Allows the 69735A Pulse Train Output card to signal the computer via the interrupt system when it has finished its last output pulse. Must be used in conjunction with the interrupt statements as described in Chapter 10 of this manual. Note if this statement is used, a subsequent Init statement will cause the interrupt to become disabled as if a Disable_intr statement was executed.
Disable_intr(Name)	Inhibits the Pulse Train Output card from signalling the computer when it's done. Meaningful only when the Enable_intr statement has already been programmed.
Set_timeout(Name,Value)	Changes the value of Timeout from the value specified in the menu to the Value specified in this statement. The legal range of Value is from 0.001 through 128,000 seconds.

69736A Timer/Pacer Card

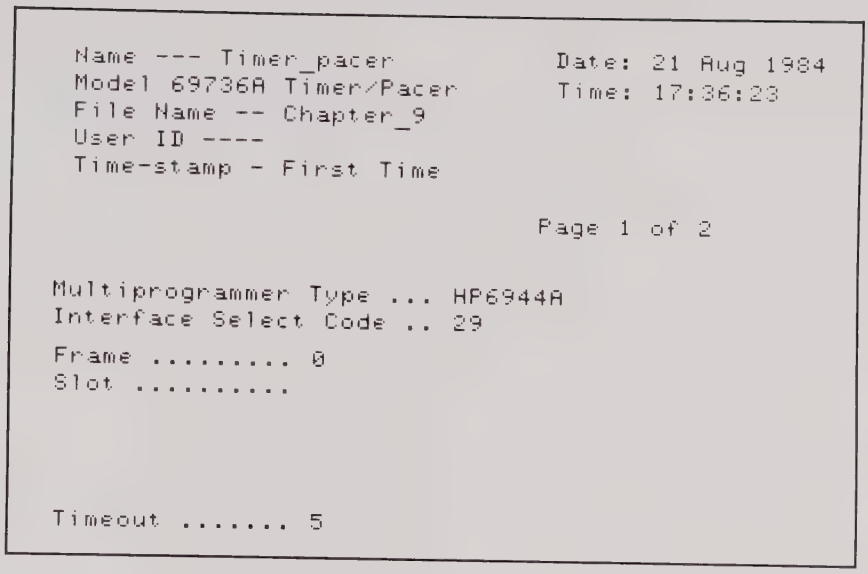
The 69736A Timer/Pacer card is a programmable timing signal generator. The card can be used in either of two modes; one shot or continuous.

In the one shot mode, the card will produce a single programmable output pulse. In the continuous mode, the card produces a square wave output having a programmable period. In both modes, data sent to the card specifies the period of the output pulse as measured from leading edge to trailing edge (See figures below). This period is programmable from 1.0 microsecond to 65535 seconds, the smallest increment being one microsecond. More information on the Model 69736A hardware is contained in its Operating Manual (HP Part No. 69736-90003).



Menu Entries For Model 69736A

Before programming the 69736A Timer/Pacer card you must specify various parameters in the Configuration Menu for the card. Following are descriptions of all the parameters required to configure this function. The parameters on the menus, as well as how to use the menu keys, are described in detail in Chapter 3.



Menu Page 1 of 2

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ..... 8
HP-IB Address ......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the Counter/Totalizer card. It has a legal range from 0 to 7. The default value is 0.

Slot

Identifies the location of the Timer/Pacer card within the frame. It has a legal range from 0 to 15. This parameter **MUST** be filled in; it has no default.

Menu Page 1 of 2 (continued)

Timeout

Sets the time (in seconds) that will be allowed for the Timer/Pacer card to complete its operation. If the Output and Wait_for statements do not detect completion within this time, a Timeout error (#803) is generated. Any number between 0.001 and 128,000 may be entered here. The default value is only 5 seconds and must be changed if you wish to program a pulse duration greater than 5 seconds with an Output statement. The Wait_for statement is used for software synchronization of the hardware functions and is discussed in greater detail in Chapter 10.

```
Name --- Timer_pacer              Date: 21 Aug 1984
Model 69736A Timer/Pacer          Time: 17:36:32
File Name -- Chapter_9
User ID ----
Time-stamp - First Time

                                  Page 2 of 2

Data Conversion ..... Standard


Run Time Limit Checking ... No


Initial Value ..... 0
Initial Mode ..... One Shot
```

Menu Page 2 of 2

Data Conversion

Specifies how the Value parameter is to be scaled by the Output and Preset statements. One of two options can be selected: Standard and User's.

1. "Standard" is the default conversion for this function. Standard units are in terms of "seconds".

2. "User" allows data to be specified in terms of the application rather than in terms of the card's Standard Units. As shown below, four additional entries appear on the menu when User Units are selected.

User's Units
P1 P2

Standard
P1 P2

Are values used only when the User Conversion parameter is selected. These parameters allow you to fill in the ratio of User's units and Standard units so the 14752A can convert the user data into information that can be recognized by the card. The default values for P1 are 0; for P2 they are 1.

More About Data Conversion

To illustrate the use of data conversion, suppose we would like to program the 69736A Timer/Pacer card in minutes rather than seconds. The card requires output values to be in terms of seconds if Standard units were used. Therefore, User conversion must be selected on menu page 2 if we expect the card to accept values in terms of minutes. When "User" is selected in the menu, four other parameters must also be specified. These parameters give the 14752A all the data it needs to scale User's units (minutes) to Standard units (seconds) before sending the value to the card. To convert minutes into

Menu Page 2 of 2 (continued)

seconds we realize that there are 60 seconds in 1 minute. Since this is a simple ratio (60 seconds/minute) the P1 parameters and the P2 parameters are entered like this:

	P1	P2
User's Units (minutes)	0	1
Standard Units (seconds)	0	60

Even though User's Units may be more convenient, it takes the 14752A software 20% to 30% longer to process the data than if Standard units were specified. However this is still faster than doing the data conversion in BASIC.

Run Time Limit Checking — If YES is selected, values sent to the Timer/Pacer card will be checked against the specified Maximum and Minimum Period. This optional parameter can be toggled on or off by selecting YES or NO. Default is NO limit checking.

High Limit — Appears only when Run Time Limit Checking is on. Specifies the upper limit for the output pulse period. An error (#802) will be generated when a programmed value is greater than this value. The value entered in the menu must be greater than the Low Limit specified. This value should be entered in the same units that are selected for programming units. This parameter **MUST** be filled in; it has no default.

Low Limit — Appears only when Run Time Limit Checking is on. Specifies the lower limit for the pulse period. An error message (#802) will be generated when a programmed value is less than this limit. The value entered in the menu must be less than the High Limit specified. This value should be entered in the same units that are selected for programming units. This parameter **MUST** be filled in; it has no default.

Initial Value — Specifies the period that will be Preset to the Timer/Pacer card when the Init or System__init statement is executed. The default value is zero (i.e. no output). This value should be entered in the same units that are selected for programming units.

Initial Mode — Specifies the initial mode programmed by an Init or System__init statement. Either the One-shot or the Continuous mode can be selected. The default is One-Shot mode.

Programming Fundamentals

Only a few fundamental statements are needed to program the Timer/Pacer card. The following is a summary of these statements. For details concerning the syntax requirements for the Name and Value parameters refer to Chapter 3

Statement	Description
Init(Name)	Loads the card with the initial period and mode values which were previously specified in the configuration menu. It also aborts any pulses currently being generated, disables interrupts, and resets Timeout to the value specified in the menu. After an Init statement, the output pulse can be started with a Start statement or external trigger signal.
Set_oneshot(Name)	Sets the Timer/Pacer card to the One Shot mode.
Set_continuous(Name)	Sets the Timer/Pacer card to the Continuous mode.
Output(Name,Value)	Sends a data value to the function and starts the pulse output. If one shot mode is selected (the default case), causes the Timer/Pacer card to generate a pulse whose width is specified by Value. If continuous mode is selected, will cause the Timer/Pacer card to generate a square wave whose period (leading edge to leading edge) is two times Value. This statement will finish executing: in one shot mode when the pulse is done, or in continuous mode when the first output pulse is done.

Example 1. Using the Timer/Pacer as a programmable one-shot.

The Timer/Pacer card can be programmed so that it will generate a single output pulse of precise duration. This accurate pulse can be used as a timing signal to another instrument, such as a counter. If we use the Timer/Pacer card output to gate the counter on and off, we can determine how many counts have occurred per unit of time (i.e. cycles/second or, frequency). This example can be easily programmed like this:

Programming Example for Gating a counter.

```
10 COM/Names/Gate_signal
20 Configure ("Ask_me","MYFILE")
30 System_init
40 ! First we would have to set up the counter
50 ! to operate in a start/stop mode; then
60 ! wire the Timer/Pacer output to the counter's
70 ! enable input.
80 !
90 ! Once that is done, we can set the mode of
100 ! operation of the 69736A.
110 !
120 Set_one_shot(Gate_signal)
130 !
140 ! Now send a 100 microsecond gate signal
150 !
160 Output(Gate_signal,100E-6)
170 !
180 ! Now the counter can be read and the data analyzed.
190 !
200 END
```

Example 2. Using the Timer/Pacer as a timebase.

There are many applications which require a signal with a precise frequency or pulse period. One such application is to "pace" the Buffered DAC function which is discussed later in this chapter. The 69736A Timer/Pacer card can be used effectively for this or any other application where a programmable square wave output is needed. The following programming example illustrates the steps involved to use the Timer/Pacer card. For the purpose of this example we will use all the default parameters in the menus.

Programming Example for a 69736A Timebase

```
10 COM/Names/Time_base
20 Configure ("Ask_me","MYFILE")
30 !
40 ! First, we will set the continuous mode
50 !
60 Set_continuous(Time_base)
70 !
80 ! Now we will generate a 50Hz square wave
90 !
100 Output(Time_base, .01)
110 !
120 ! Now we can change the frequency of
130 ! the output to 100Hz.
140 !
150 Output(Time_base, .005)
160 !
170 ! If variables are used, the frequency can
180 ! be programmed as a result of a calculation.
190 !
200 Frequency = 500
210 Pulse_width = 1/(Frequency *2)
220 Output (Time_base,Pulse_width)
230 !
240 ! To stop the output...
250 !
260 Init(Time_base)
270 !
280 ! To start it again with a 500kHz output
290 !
300 Set_continuous(Time_base)
310 Output(Time_base,1E-6)
320 END
```

Advanced Programming

There are 14752A statements which allow special purpose programming of the Timer/Pacer card. They are considered advanced programming statements because you need a good understanding of the Multiprogrammer I/O cards and system to put together a meaningful program. In general, these statements are used only when precise control of the timing is a concern. They allow features such as: parallel outputs from more than one function card, external triggering, interrupts, and checking status. All of these features are explained in greater detail in Chapter 10. The following is a list of the advanced programming statements which can be used for the 69736A Timer/Pacer card.

Statement	Description
Preset(Name,Value)	Sends a pulse period value to the function but does not start the output pulse. Used as a set-up operation for a Start statement or External Trigger. Finishes quickly because it does not have to wait for a response from the card.
Start(Name1,..,Name8)	Starts the pulse output with the period specified by the last Value sent to the card. If more than one Name is listed, all the named functions start at the same instant. This statement does not wait for the output pulse to complete therefore it finishes quickly. That is, with the Start statement the program will continue even if the pulse has not yet finished.
Wait_for(Name)	Program execution is paused until the function has completed its operation or a timeout occurs. Used if you want to wait for the Timer/Pacer card to finish when it has been started with a Start command or an External Trigger signal. In the Continuous Mode, program execution waits for the first complete squarewave to occur.
Check_done(Name,Flag)	Inquires about the status of the function's operation. Flag is set to a one if the output pulse has completed. Once a value of one is returned, the Check_done statement will subsequently return zero until the function is Started again.
Enable_intr(Name,Vector)	Allows the 69736A Timer/Pacer card to signal the computer via the interrupt system when it has finished its output pulse. Must be used in conjunction with the interrupt statements as described in Chapter 10 of this manual. Note, if this statement is used, the Init statement will cause the interrupts to become disabled as if a Disable_intr statement was executed.
Disable_intr(Name)	Inhibits the Timer/Pacer function from signaling the computer when it's done. Meaningful only when the Enable_intr statement has already been programmed.
Set_timeout(Name,Value)	Changes the value of Timeout from the value specified in the menu to the value specified by "Value". The range of legal values is from 0.001 and 128,000 seconds.

69775A Counter/Totalizer Card

The 69775A Counter/Totalizer card counts events that are represented by electrical signals applied to separate up-count and down-count inputs of the card. The card's counter has a 1 MHz maximum count rate and a 16-bit (0 to 65535) capacity. The card is designed to be able to count up and down simultaneously. Even when up-count and down-count pulses occur at the same instant, they are counted properly at rates up to the card's 1 MHz maximum.

In addition there are also separate enable inputs for both the up-count and the down-count inputs. These inputs can be used as a gating function to allow the counter card to accept inputs only during a specific time window.

The counter also has the ability to be loaded or "preset" with a data value. This value is used as the initial value for the counter circuits. A pulse applied to the up-count input will increment the counter starting from this value; a pulse input to the down-count input will decrement the counter starting from this value. If the default data type (Unsigned binary) is used and the counter reaches its maximum count of 65535, an additional pulse to the up-count input will cause the count value to reset to zero or "overflow". Similarily, if the count value is at zero and another pulse is applied to the down-count input the count value will be set to the maximum count of 65535. This is called an "underflow". Since both the "overflow" and the "underflow" conditions are detected by the card, exactly which one occurs is dependent on the actual application. Overflow and underflow are conditions that are used with the Wait_for, Check_done, and Enable_intr statements when using the "Unsigned binary" data type.

If the 2's complement data type is selected using the menu, the overflow and underflow conditions are not used with the Wait_for, Check_done, and Enable_intr statements. Instead there is another condition, "near zero", that is used with these statements. The near zero condition will indicate when the Counter/Totalizer card either counts down from zero to -1 or counts up from -1 to zero.

The card has analog comparators on its input circuits for interfacing flexibility. When the card is shipped, these inputs have a $+1.2$ volt threshold, making them suitable for operation at TTL levels. This threshold can be changed by changing a number of hardware jumpers on the card. Refer to the Operating Manual (HP Part No. 69775-90003) for more information about the card's hardware.

Menu Entries For Model 69775A

Before programming the 69775A Counter/Totalizer card, you must specify various parameters in the Configuration Menu for the card. Following are descriptions of all the parameters required to configure this function. Instructions on how to use the menus and additional information on most of the parameters are given in Chapter 3. The Counter/Totalizer function's parameters are described as follows:

```
Name --- Counter                    Date: 21 Aug 1984
Model 69775A 16 Bit Counter         Time: 17:37:03
File Name -- Chapter_9
User ID ----
Time-stamp - First Time

                                    Page 1 of 2

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot ..........



Timeout ....... 5
```

Menu Page 1 of 2

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ..... 8
HP-IB Address ......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the Counter/Totalizer card. It has a legal range from 0 to 7. The default value is 0.

Slot

Identifies the location of the Counter/Totalizer card within the frame. The legal range is from 0 to 15. This parameter **MUST** be filled in; it has no default value.

Menu Page 1 of 2 (continued)

Timeout

This parameter is only used when the Wait_for statement is programmed. Timeout sets the length of time (in seconds) that will be allowed for the Counter/Totalizer card to either:

1. overflow or underflow (when using the unsigned binary data type)

   or,

2. count down from 0 to −1 or count up from −1 to 0, (when using the 2's complement data type).

If the Wait_for statement does not detect either of these conditions before the Timeout value has expired, an error message (#803) is generated. Any number between 0.001 and 128,000 may be entered for Timeout. The default value is only 5 seconds. The Wait_for statement is described in the Advanced Programming section for this card.

```
Name --- Counter                    Date: 21 Aug 1984
Model 69775A 16 Bit Counter         Time: 17:37:12
File Name -- CHAPTER_9
User ID ----
Time-stamp - First Time

                                  Page 2 of 2

Standard Data Type ........ Unsigned Binary
Data Conversion ........... Standard


Run Time Limit Checking ... No


Initial Count ............. 0
```

Menu 2 of 2

Standard Data type — Specifies either "Unsigned Binary" or "2's Complement Binary" as the Standard Data Type for the Counter/Totalizer card. The default selection is Unsigned Binary.

Unsigned Binary — Allows you to read or send only positive numbers. The 14752A Output, Preset, and Input statements interpret these numbers to be unsigned binary values for the Counter/Totalizer card. The valid range of numbers is from 0 to 65535.

2's Complement — Allows you to read or send positive or negative numbers from −32768 to 32767. The 14752A Input, Output, and Preset statements interpret the negative numbers as 2's complement binary values for the Counter/Totalizer card.

NOTE

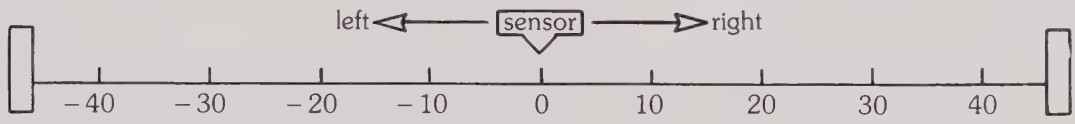
When using 2's complement data type, there is NO indication of either the overflow (a count greater than 32767) or underflow (a count less than −32768) condition. There is, however, an indication of a "near zero" condition. Refer to the Advanced Programming section for more details on using the "near zero" condition with Check_done, Wait_for, and Enable_intr statements.

Example using 2's complement data type:

Suppose we have a position sensor that generates pulses on one output to indicate movement to the left and pulses on a second output to indicate movement to the right. Let's connect the "left indicator"

Menu Page 2 of 2 (continued)

to the count down input of the Counter/Totalizer, and the "right indicator" to the count up input of the Counter/Totalizer card. Now, if we select 2's complement data type in the menu, the count value we read from the card will give us the relative distance of the position sensor from the center. Negative readings mean the sensor is positioned to the left of center; Positive readings mean the sensor is positioned to the right of center. Since our sensor cannot move a distance to the left or right that exceeds the range of 2's complement numbers (−32768 to 32767) we don't have to worry about the counter overflowing or underflowing. If we did exceed the range we would get unexpected results.



Data Conversion

This parameter specifies how the data is to be scaled for the Input, Output, and Preset statements. One of two data conversion options can be selected: Standard and User.

"Standard" conversion for this card means that the data values in an Input statement are returned to the computer in the form of unsigned binary or 2's complement values, whichever was selected as the standard data type. Likewise, data values sent in an Output, Preset, or Init statement are sent to the card in the form of unsigned binary or 2's complement numbers, whichever was selected as the standard data type. Standard is the default data conversion for this function.

"User" allows data to be specified in terms of the application rather than in terms of the card's Standard Units. As shown below, four additional entries appear on the menu when "User" conversion is selected.

User's
P1 P2

Standard
P1 P2

Are values used only when the User Conversion option is selected. These parameters allow you to fill in the ratio of User Units and Standard Units values so the 14752A can convert the user data into information that can be recognized by the card. The default values for P1 are 0; P2 default values are 1.

More About Data Conversion

To further illustrate the use of data conversion, assume that a brewery wants to determine the number of cases of beer which come off a conveyor which feeds the beer bottles into the cases. The 69775A Counter/Totalizer card together with some external hardware can count the number of bottles. This would be the Standard Units; the number read from the Counter/Totalizer would be divided by the number of bottles per case. However, if User Units were selected the data read from the Counter card would be converted from the number of bottles to the number of cases by the 14752A.

```
Name --- Counter                    Date: 21 Aug 1984
Model 69775A 16 Bit Counter         Time: 17:37:20
File Name -- Chapter_9
User ID ----
Time-stamp - First Time

                                  Page 2 of 2

Standard Data Type ........ Unsigned Binary
Data Conversion ........... Standard


Run Time Limit Checking ... No


Initial Count ............. 0
```

Menu Page 2 of 2 (continued)

From this example it can be seen that Standard conversion is used when you want to program a function by using numbers it expects. In number of counts in our case above. Standard conversion is also 20% to 30% faster then with User's conversion although User's Conversion provides a much easier way to get data from a function.

When User's conversion is selected in the menu, four other parameters must also be specified. These parameters give the 14752A all the data it needs to perform the necessary conversion calculations. For example, if the there are 24 bottles in a case of beer this would specify one point for the conversion. Another point would be the zero counts being 0 cases. This yields a range of values as follows:

	minumum input (P1)	maximum input (P2)
User's Units	0 cases	1 case
Standard Units	0 counts	24 counts

From this list we can see that:
User's P1 = 0
User's P2 = 1
Standard P1 = 0
Standard P2 = 24

Run Time Limit Checking If YES is selected, values sent to the Counter/Totalizer card by the Output and the Preset statements will be checked against the specified High and Low Limits. This optional parameter can be toggled on and off by selecting YES or NO. The default is NO limit checking. Note, only values sent to the card are checked, not values read from the card.

Menu Page 2 of 2 (continued)

High Limit

Is used only when the Run Time Limit Checking is ON. Specifies the upper limit for the count value. The value entered must be greater than the Low Limit specified. An error message (#802) will be generated when a programmed value is greater than the High Limit. This value should be entered in the same units that are selected for the programming units. Note, if Run Time Limit Checking is ON, this parameter **MUST** be filled in; it has no default.

Low Limit

Is used when Run Time Limit Checking is ON. Specifies the lower limit for the count value. The value entered must be less than the High limit that is specified. An error message (#802) will be generated when a programmed value is less than this limit. The value should be entered in the same units that are selected for the programming units. If Run Time Limit Checking is ON then this parameter **MUST** be filled in; it has no default.

Initial Count

Specifies a count value which will be loaded into the card when the Init or System_init statements are executed. This value should be entered in the same units that are selected for programming units. The default value is 0. It also should be between the High Limit and Low Limit values if Run Time Limit Checking is selected.

Programming Fundamentals

Only a few fundamental statements are needed to program the Counter/Totalizer card. The following is a summary of these statements. For details concerning the syntax requirements for the Name and Value parameters, refer to Chapter 3.

Statement	Description
Init(Name)	Loads the card with the Initial Count Value specified in the Configuration Menu and, if the one-shot mode of operation is being used, Init will clear the input enable circuits. Init also disables interrupts, and resets Timeout to the value specified in the menu.
Input(Name,Value)	Reads the contents of the card's counters and returns the number in the variable, Value. Value is in the format specified by the Data Type and Data Conversion selected in the menu.

Example #1

The 69775A Counter/Totalizer can be used to measure the rotational speed of a motor. In this example, the motor's shaft speed is detected by a shaft encoder that generates one pulse for each revolution of the motor. The output of the encoder is connected to the Counter's up-count input. The number of pulses counted by the Counter/Totalizer card will be proportional to the motor's shaft speed. An additional Multiprogrammer card, the 69736A Timer/Pacer card, is used to supply an enable signal for the Counter card. This signal provides a precise timing reference for the measurement. The following example illustrates how to program the 69775A Counter/Totalizer card to measure the shaft speed of a motor.

The Counter card is identified with the Name, "Shaft_speed" and the Timer/Pacer card with the Name, "Timebase", during the Configuration phase using the menus. We will use default values for both cards. These default values will set the Counter card's counters to zero and set the Timer/Pacer card to the one shot mode. The Timeout parameter for the 69736A Timer/Pacer card will be increased to 100 seconds to allow for the long pulse duration. A sample program for this example is on the next page.

Programming Example for Motor Speed Measurement

```
10 COM/Names/Shaft__speed,Timebase
20 Configure("Ask__me","MYFILE")
30 !
40 ! First, let's initialize the Counter card
50 ! to zero the count registers
60 !
70 Init(Shaft__speed)
80 !
90 ! Let's also initialize the Timer/Pacer card
100 ! thereby setting it to the one-shot mode.
110 !
120 Init(Timebase)
130 !
140 ! Let's assume that the motor is now turning.
150 ! We can generate a single pulse with a pulse width
160 ! of 60.0 seconds.
170 !
180 Output(Timebase,60.0)
190 !
200 !
210 ! The number of pulses can be read from the
220 ! Counter card directly in terms of
230 ! revolutions per minute.
240 !
250 Input(Shaft__speed,Speed)
260 END
```

Advanced Programming

There are 14752A statements which allow special purpose programming of the Counter/Totalizer function. They are considered advanced programming statements because you need a good understanding of the Multiprogrammer I/O cards and system to put together a meaningful program. In general, these statements are used only when precise control of the card's counting is a concern. They allow features such as: starting multiple Names in parallel, interrupts, and checking status. All of these features are explained in greater detail in Chapter 10. The following is a list of the advanced programming statements which can be used for the 69775A Counter/Totalizer function.

Statement	Description
Output(Name,Value)	Sends a count value to the function and loads it into the counters on the Counter/Totalizer card.
Preset(Name,Value)	Sends a count value to the function but does not load it into its counters. Used as a set-up operation for the Start statement or an external trigger signal.
Start(Name1,...,Name8)	Loads the count value, that was previously sent to the card with the Preset statement, into the card's counters. If more than one Name is listed, all the named functions start in parallel. This statement will also reset the Counter card if it is in the One-shot Mode. (The card modes are hardware configured and are not programmable. Refer to the 69775A Operating Manual (HP P/N 69775-90003) for further information.)
Wait_for(Name)	Program execution is suspended until either the Counter/Totalizer card overflows or underflows when using the unsigned binary data type or the "near zero" condition occurs when using the 2's complement data type. If the conditions do not occur within the time specified for timeout, an error message (code 803) will be generated. Used if you want to suspend your program execution until either condition occurs.
Check_done(Name,Flag)	Inquires about the status of the function's operation. Flag is set to a one if either the count has overflowed or underflowed when using the unsigned binary data type or the "near zero" condition occurs when using the 2's complement data type. Once a value of one is returned, the Check_done statement will subsequently return zero until either condition re-occurs.
Enable_intr(Name,Vector)	Allows the 69775A Counter/Totalizer card to signal the computer via the interrupt system when either it has overflowed or underflowed when using the unsigned binary data type or, the "near zero" condition occurs when using the 2's complement data type. Enable_intr must be used in conjunction with the BASIC statement ON INTR and the 14752A statement Enable_sc_intr as described in Chapter 10 of this manual. Note: if this statement is used, the Init statement will cause the interrupts to become disabled as if a Disable_intr statement was executed.

Statement	Description
Disable_intr(Name)	Inhibits the Counter/Totalizer card from signalling the computer as just described. This statement is meaningful only after Enable_intr has been programmed.
Set_timeout(Name,Value)	Changes the value of Timeout from the time specified in the menu to Value. Time is specified in seconds, from 0.001 and 128,000.

Advanced Programming Example

The same problem we explained in the section "More About Data Conversion" can be solved using Output and Wait_for statements instead of User Conversion. Again, suppose the brewery wants to know when a total of 24 bottles of beer roll past a point on the conveyor. The Counter card can be preset to a value that will cause the Counter to overflow after a total of 24 counts which represent the 24 bottles of beer. This value is calculated by subtracting the desired count, in this case 24, from the Counter's overflow value of 65536. The program example listed below will initialize the Counter card and then load the desired count value using the Init and Output statements, respectively. The Wait_for statement is used for signalling the program when the counter overflows and 24 bottles of beer pass a point on the conveyor.

Programming Example for Program Signalling of Hardware Event

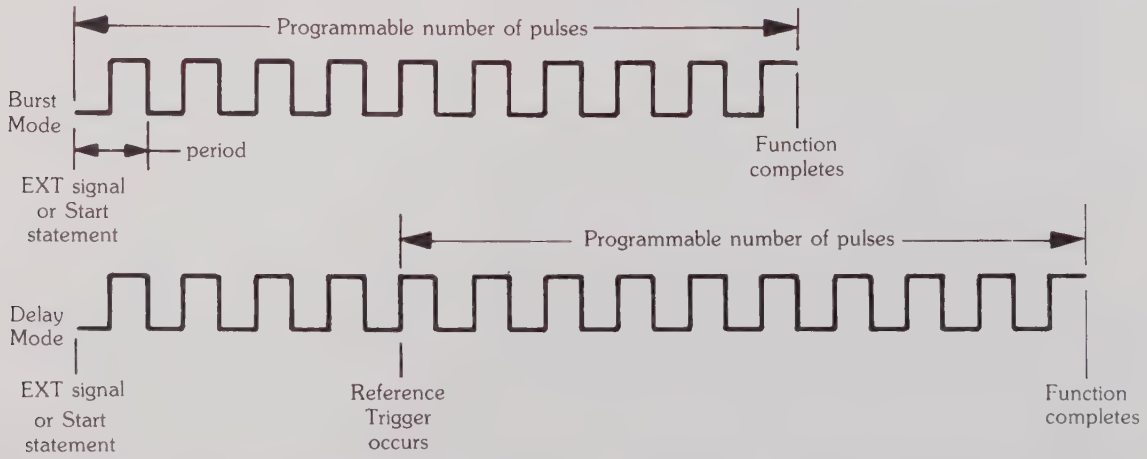
```
10 COM/Names/Bottle_count
20 Configure("Ask_me","MYFILE")
30 !
40 ! The value to be loaded onto the
50 ! Counter card is calculated.
60 !
70 Bottle_per_case = 24
80 Max_count = 65536
90 Ref_count = Max_count - Bottle_per_case
100 !
110 ! The Counter card is initialized
120 !
130 Init(Bottle_count)
140 !
150 ! The reference count value calculated will now be
160 ! loaded onto the Counter card
170 !
180 Output(Bottle_count,Ref_count)
190 !
200 ! The program will now wait for the 24 bottles
210 ! to be counted before proceeding
220 !
230 Wait_for(Bottle_count)
240 END
```

Timer Function

The Timer function uses a Model 69736A Timer card to generate a continuous square wave output with a programmable period between 2 microseconds and 131,070 seconds. The output pulse train can be started with an External Trigger signal (EXT) or a Start statement. The Timer function has two modes of operation: Burst and Delay. The Burst mode will generate a programmable number of pulses and then stop. The Delay mode will generate a continuous stream of pulses that can be stopped by either an external signal (called Reference Trigger) or a Stop statement. The Stop statement will cause the pulses to stop immediately. When using the Reference Trigger, the output pulse train can be stopped immediately or after a programmable delay. See Figure below.

The Reference Trigger input signal can be qualified by either an on-card switch (S2) or an optional 69731B Digital Output card. Qualifying the Reference Trigger allows programmable selection of either Burst or Delay mode in addition to programmable selection of either positive or negative true Reference Trigger input signals. When the switch is used to qualify the Reference Trigger, S2-A (open or closed) determines the logic level (positive or negative) of the input signal. Whether S2-D is open or closed determines the sense (edge or level) of the Reference Trigger input. Note that switch S2-D must be closed and S2-A opened for Burst mode operation and S2-D open for Delay mode operation. If a Digital Output card is used to qualify the Reference Trigger input, all switches should be opened except S2-F allowing the level and sense to be determined by the statements: Set_reftrg_pos, Set_reftrg_neg, Set_burst_mode, and Set_delay_mode. These statements are described later in this section. The programmable delay is accomplished by counting the output pulses with a Model 69775A Counter card. When using one Counter card up to 32767 pulses can be counted before stopping the pulse train. One more 69775A Counter card can be added to the Timer function to allow up to 2,147,385,345 pulses to be counted. This function is considered complete when the last programmed pulse has finished.

Figure C-5 in Appendix C shows the required I/O card interconnections for the Timer Function. More information (card output pins, switch settings etc.) on the Multiprogrammer I/O cards used for this function is in their Operating Manuals. The HP part number for the Model 69736A's manual is 69736-90003. The HP part number for the Model 69775A's manual is 69775-90003. The HP part number for the Model 69731B's manual is 69731-90003.



Note: Reference Trigger can occur at any time, but may take up to one pulse period to take effect. Make sure that only one transition of Reference Trigger occurs. If you get a second transition, the Timebase pulse train will continue forever.

Timer Function Burst and Delay Mode

Menu Entries for Timer

Before programming the Timer function you must specify various parameters in the Configuration Menu for the card. Following are descriptions of all the parameters required to configure this function. Most of the parameters on the menus, as well as how to use the menu keys, are described in detail in Chapter 3.

```
Name --- Timer                    Date: 21 Aug 1984
Model Timer                       Time: 17:34:53
File Name -- Chapter_9
User ID ----
Time-stamp - First Time
                                  Page 1 of 3

Multiprogrammer type ... HP6944A
Interface Select Code .. 29

Timer/Pacer Frame .. 0
Timer/Pacer Slot ...
Counter Frame ..... 0
Counter Slot .......

Counter size ... 15 Bits


Timeout ........ 5
```

Menu Page 1 of 3

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ..... 8
HP-IB Address ......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Timer/Pacer Frame

Specifies the Multiprogrammer frame which contains the 69736A Timer/Pacer card. It has a legal range from 0 to 7. The default value is 0.

Timer/Pacer Slot

Identifies the location of the 69736A Timer/Pacer card within the frame. It has a legal range from 0 to 15. This parameter **MUST** be filled in; it has no default.

Menu Page 1 of 3 (continued)

Counter Frame

Specifies the Multiprogrammer frame which contains the 69775A Counter card. It has has a legal range from 0 to 7. The default is 0.

Counter Slot

Identifies the location of the 69775A Counter card within the frame. It has a legal range from 0 to 15. This parameter **MUST** be filled in; it has no default value.

Counter Size

If you have one 69775A Counter card, select "15 Bits". If you have two 69775A counter cards, select "31 Bits". If you have two Counter cards, the following two menu entries must be filled in.

Extension Counter

Appears only if Counter Size is 31 bits. Specifies the Multiprogrammer frame which contains the additional Counter card. It has a legal range from 0 to 7. The default is 0.

Extension Counter Slot

Appears only if Counter Size is 31 bits. Identifies the location of the additional 69775A Counter card within the frame. It has a legal range from 0 to 15. This parameter **MUST** be filled in; it has no default

Timeout

Sets the length of time (in seconds) that will be allowed for the Timer function to complete its operation. If the Wait_for statement does not detect completion within this time, a Timeout error message (code #803) is generated. Any number between 0.001 and 128,000 may be entered here. The default value is 5 seconds. The Wait_for statement is used for software synchronization of the hardware functions and is discussed in greater detail in Chapter 10. Note that the maximum value 128,000 does not cover the maximum delay time that is required for the function to complete an operation involving a delay greater than 128,000 seconds. if a delay greater than 128,000 seconds is required, then you must not use the Wait_for statement.

```
Name --- Timer                    Date: 21 Aug 1984
Model Timer                       Time: 17:35:11
File Name -- Chapter_9
User ID ----
Time-stamp - First Time

                                  Page 2 of 3

Standard Data Type ........ Unsigned Binary
Data Conversion ........... Standard


Run Time Limit Checking ... No
```

Menu Page 2 of 3

Data Conversion

Specifies how the period value is to be scaled by the Set_period and Init statements. One of two options can be selected: Standard and User's.

1. "Standard" is the default conversion for this function. Standard units are in terms of 'seconds'.

2. "User" allows data to be specified in terms of the application rather than in terms of the card's Standard Units. As shown below, four additional entries appear on the menu when User Units are selected.

User's Units P1 and P2

Standard P1 and P2

Are values used only when the User Conversion parameter is selected. These parameters allow you to fill in the ratio of User's units and Standard units so the 14752A can convert the user data into information that can be recognized by the card. The default values for P1 are 0; for P2 they are 1.

More About Data Conversion

To illustrate the use of proramming units, suppose we would like to program the Timer in milliseconds rather than seconds. The card requires output values to be in terms of seconds if Standard units were used. Therefore, User conversion must be selected on menu page 2 if we expect the card to accept values in terms of milliseconds. When "User" is selected in the menu, four other parameters must also be specified. These parameters give the 14752A all the data it needs to scale User's units (milliseconds) to Standard units (seconds) before sending the value to the card. To convert milliseconds into seconds we realize that there are 1000 milliseconds in one second. Since this is a simple ratio (1000 milliseconds/second) the P1 parameters are zero and the ratio numerator and denominator are entered into the parameters like this:

	P1	P2
User's Units (milliseconds)	0	1000
Standard Units (seconds)	0	1

Even though User's Units may be more convenient, it takes the 14752A software 20% to30%longer to process the data than if Standard units were specified. However this is still faster than doing the data conversion in BASIC.

Run Time Limit Checking — If YES is selected, values sent to the Timer function will be checked against the specified Maximum and Minimum Period and max/min Count limits. This optional parameter can be toggled on or off by selecting YES or NO. The default is NO limit checking.

Maximum Count — Is used only when Run Time Limit Checking is ON. Specifies the upper limit for the count. The value entered in the menu must be greater than the Minimum Count specified. An error (#802) will be generated when a programmed value is greater than this limit. The value must be less than 32,767 if one Counter card is used and less than 2,147,385,345 if two Counter cards are used. The default is 32,767.

Minimum Count — Is used only when Run Time Limit Checking is ON. Specifies the lower limit for the count. An error (#802) will be generated when a programmed value is less than this limit. The value entered in the menu must be less than the Maximum Count specified and greater than or equal to zero. The default is zero.

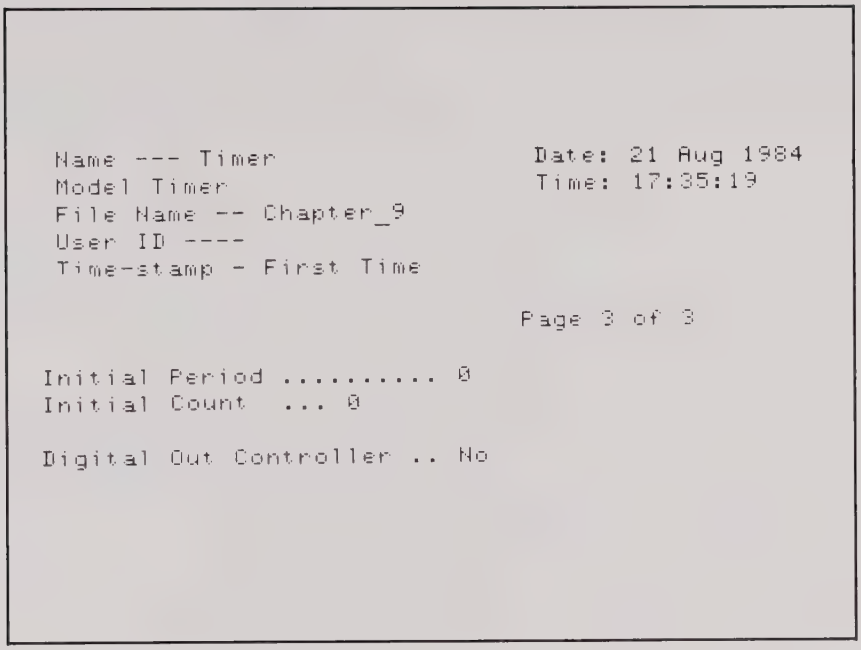
Menu Page 2 of 3 (continued)

Maximum Period

Is used only when Run Time Limit Checking is ON. Specifies the upper limit for the period data sent to the Timer card. An error (#802) will be generated when a programmed data value exceeds this limit. The value entered in the menu must be greater than the Minimum Period specified. The default value for this parameter is the maximum programmable period of 131,070 seconds.

Minimum Period

Is used only when Run Time Limit Checking is ON. Specifies the lower limit for the period data sent to the Timer card. An error(#802) will be generated when a programmed data value is less than this limit. The value entered in the menu must be less than the Maximum period specified and greater than zero. The default value is the minimum programmable period of two microseconds.



Menu Page 3 of 3

Initial Period

Specifies the period (in seconds) of the output pulse train when Init or System_init statement is executed. It has a legal range from 2 microseconds to 131,070 seconds. The default value is 100 milliseconds.

Initial Count

Specifies a value to be preset to the Counter card by the Init or System_init statements. In Burst mode, Initial Count specifies the number of pulses generated. In Delay mode it specifies the number of pulses that will occur after the Reference Trigger is received. It has a legal range from 0 to the maximum count. The maximum count is 32767 when one Counter card is used and 2,147,385,345 if two Counter cards are used. The default value is 0.

Digital Out Controller

If you have a 69731B Digital output card, enter YES. If you don't have one, enter NO. If YES is selected the following four menu entries appear.

Controller Frame

Specifies the Multiprogrammer frame which contains the 69731B Digital Output card. It has a legal range from 0 to 7. The default value is 0.

Menu Page 3 of 3 (continued)

Controller Slot

Identifies the location of the 69731B Digital Output card within the frame. It has a legal range from 0 to 15. This parameter **MUST** be filled in; it has no default.

Timer Mode

Selecting "Delay" allows edge-sensitive Reference triggers (used for Delay mode operation). Selecting "Burst" makes the Reference Trigger input level sensitive (used for Burst mode operation). This initial condition is used only if you are using a 69731B Digital Output Controller card.

Reference Trigger Polarity

Selecting Positive makes the logic sense of Reference Trigger positive true. Selecting negative makes the logic sense negative true. This initial condition is used only if you are using a 69731B Digital Output Controller card.

Programming Fundamentals

Only a few fundamental statements are needed to program the Timer function. The following is a summary of these statements. For details concerning the syntax requirements for the Name and Value parameters, refer to Chapter 3.

Statement	Description
Init(Name)	Loads the Timer card with the initial period value and the Counter card(s) with the initial count value as specified in the configuration menu and then starts the pulse train. It also aborts any pulse train currently being generated, disables interrupts, and resets Timeout to the value specified in the menu.
Start(Name1,..,Name8)	Starts the output pulse train with the period value specified by the last Value sent to the card. If more than one Name is listed, all the named functions start in parallel and at the same instant. All of the Names must be located at the same Multi-interface. This statement does not wait for the Timer function to complete, therefore it finishes quickly. That is, with the Start command the program will continue even if the pulse train has not finished.
Stop(Name)	Aborts the pulse train currently being generated. The period value is not affected. Set_period(Name,Value) Sets the output pulse period. The acceptable range for Value is .000002 to 131,070 seconds. The smallest increment that can be programmed is two microseconds.
Set_count(Name,Value)	In Burst mode, sets the number of output pulses that will be generated when the Start statement is executed. A new Value can be sent at any time (even while a pulse train is in progress). However, if this statement is issued any time after a Start statement (and a Stop statement has not been issued) a burst of pulses of the specified count and period will be generated. In Delay mode, sets the number of output pulses that will occur after the Reference trigger is received. A new Value can be sent at any time. However, if this statement is issued after the receipt of a Reference Trigger, the specified Value becomes the new delay count and that many pulses will be generated before stopping. The range of Values is between 0 and 32767 if one counter card is used, and between 0 and 2,147,385,345 if two Counter cards are used.
Set_period(Name, Value	Sets the output pulse period. The acceptable range for Value is .000002 to 131,070 seconds. The smallest increment that can be programmed is two microseconds.

Example 1. Using the Timer in Burst Mode.

There are many applications which require a programmable pulse train with a precise frequency or pulse period. A timebase is a device which produces a signal having a very precise frequency or pulse period. The Timer function can be used effectively as a timebase because of its crystal-controlled pulse period. The output pulses can be used to trigger an instrument at a known rate or to externally trigger another Multiprogrammer function card. One such application is to "pace" the Buffered DAC function which is discussed in the "Buffer" section of this chapter. The Timer can be used effectively for this or any other application where a programmable square wave output is needed. The following programming example illustrates the steps involved to use the Timer function to generate 4000 pulses. In the example, assume that the Timer has been assigned to the name Timer, and the default menu entries were selected in the configuration menu.

Example Program for Burst Mode Timer

```
 10 COM/Names/Timer
 20 Configure ("Ask_me","MYFILE")
 30 !
 40 ! First, we will set the pulse period to 1 ms.
 50 !
 60 Set_period(Timer,0.001)
 70 !
 80 ! Now we will set the desired number of pulses
 90 !
100 Set_count(Timer,4000)
110 !
120 ! Now to generate the burst
130 !
140 ! Start(Timer)
150 !
160 ! Now we can generate another burst
170 ! but with a 60 second period
180 !
190 Set_period(Timer,60)
200 Set_Count(Timer,4000)
210 ! If we don't want to wait 4000 minutes
220 ! we can stop the output
230 Stop(Timer)
240 !
250 ! To start it again with a 500kHz output
260 !
270 Set_period(Timer,2E-6)
280 Start(Timer)
290 END
```

Example 2. Using the Timer in Delay mode.

The Buffered A/D function, discussed in the "Buffer" section of this Chapter, is a good example of a function which can be externally triggered by the Timer function. Briefly, the Buffered A/D takes a voltage reading and places it in a memory buffer every time it is triggered. The Timer can supply the Buffered A/D function with that trigger (ie. the timing signal that initiates the data conversion). The Buffered A/D is very accurately paced because of the crystal controlled oscillator on the 69736A Timer/Pacer card. The Timer Delay mode can be used to allow the buffer to store a programmed number of readings after an event occurs. If the buffer is in Recirculate mode, we can store data before and after the event. The number of data conversions, and the time interval between data conversions, are the two key parameters for the timebase. The number of data conversions is specified by the Set_count statement and the time interval is specified by the Set_period statement.

In the following programming example assume that the Timer has been assigned to the name Timer, and the default menu entries were selected in the configuration menu.

Example Programming for Delay Mode Timer

```
 10 COM/Names/Timer
 20 Configure("Menu_off","MYFILE")
 30 !
 40 ! First set up the period to 50
 50 ! microseconds between voltage samples.
 60 !
 70 Set_period(Timer,50E-6)
 80 !
 90 ! Next, set the delay count
100 ! to take 2000 readings after Reference Trigger.
110 !
120 Set_count(Timer,2000)
130 !
140 ! Now we can start the continuous output
150 ! so the Memory card takes continuous readings
160 !
170 Start(Timer)
180 !
190 ! Once the Reference Trigger occurs, there
100 ! are only 2000 more pulses generated.
110 ! This allows a 4k memory to store 2000 readings
120 ! before Reference Trigger and 2000 readings
130 ! after Reference Trigger.
140 !
150 END
```

Advanced Programming

There are 14752A statements which allow special purpose programming of the Timer function. They are considered advanced programming statements because you need a good understanding of the Multiprogrammer I/O cards and system to put together a meaningful program. In general, these statements are used only when precise control of the timing is a concern. They allow features such as: external triggering, checking status, and interrupts. All of these features are explained in greater detail in Chapter 10. The following is a list of the advanced programming statements which can be used for the Timer function.

Statement	Description
Wait_for(Name)	Program execution is paused until the function has completed its operation or a timeout occurs. Used if you want to wait until the pulse train stops.
Check_done(Name,Value)	Inquires about the status of the function's operation. Value is set to a one if the pulse train has stopped. Once a value of one is returned, the Check_done statement will subsequently return zero until the function Starts and completes again.
Enable_intr(Name,Vector)	Allows the Timer function to signal the computer via the interrupt system when it has finished its last output pulse. Must be used in conjunction with the interrupt techniques as described in Chapter 10 of this manual. Note, if this statement is used, a subsequent Init statement will cause the interrupts to become disabled as if a Disable_intr statement was executed.
Disable_intr(Name)	Inhibits the Timer Function from signaling the computer when it's done. Meaningful only when the Enable_intr statement has already been programmed.
Set_burst_mode(Name)	Makes the Reference Trigger input level sensitive. This statement can only be used if you have a 69731B card cabled into the Timer Function. If you don't then Error 854 is generated.
Set_delay_mode(Name)	Allows edge-sensitive Reference Triggers. This statement can only be used if you have a 69731B card cabled into the Timer Function. If you don't then Error 854 is generated when this statement is executed.

Statement	Description
Set_reftrg_pos(Name)	Makes the logic sense of Refference Trigger positive true. This statement can only be used if you have a 69731B card cabled into the Timer Function. If you don't then Error 854 is generated when this statement is executed.
Set_reftrg_neg(Name)	Makes the logic sense of the Reference Trigger input negative true. This statement can only be used if you have a 69731B card cabled into the Timer Function. If you don't then Error 854 is generated.
Set_timeout(Name,Value)	Changes the value of Timeout from the value specified in the menu to the Value specified in this statement. The legal range of Value is from 0.001 and 128,000 seconds.

69776A Interrupt Card

The 69776A Interrupt card compares a programmed 16-bit reference word with an external input word applied to the card and stores the input word when a predetermined relationship exists between the two words. The comparison condition that latches the input word is called the true condition. One or more bits of the 16-bit input word can be removed from the comparison, if desired, by programming a mask word on the card. The present input word, and the reference word may be read back to the computer.

One application of the card is to perform comparison operations on binary weighted words up to 16 bits in length. The four relationships or comparison modes that can be programmed are when the input word becomes "equal to", "unequal to", "greater than", or "less than" the reference word. Once the reference word, comparison mode, and mask word have been programmed, either a start statement or an External Trigger signal can be used to enable the card to begin a new comparison. The card can be interrogated to determine if the true condition has occurred. When the comparison condition is satisfied, and if an Enable_intr statement has been issued, the 69776A card will not only store the input word that cause the true condition, but will also interrupt the computer.

The 69776A card can also be used to detect a change on any of its sixteen input bits. Although the card does not support truly independent interrupt bits, special "bit oriented" statements are provided by the 14752A to simplify using the card in this type of application. The Interrupt card can be programmed to interrupt the computer if any of the input bits change their value. This is referred to as the "Bit Interrupt" configuration. Once an interrupt is generated, the interrupting bit can be determined by either of two methods. The first way is to read the interrupting word into the computer and (using the binary functions supplied by BASIC) interrogate the value to determine which bit(s) changed. The second way is to examine a variable, called Reason, that is passed to the user's interrupt handling routine by Inter_handler. There is a more detailed explanation in Chapter 10 on how to handle interrupts.

Each of the input bits on the 69776A card has a filter to suppress input noise. These filters cause the card to ignore all input signals lasting less than 6 microseconds and allow all those longer than 12 microseconds to be stored. For signals lasting between 6 and 12 microseconds, the word may, or may not, be stored. The filter delay time can be varied between 5 microseconds and 1 second by changing components on the card as required by the application. For complete information concerning the card's hardware features refer to the 69776A Operating Manual (HP Part No. 69776-90003).

Once an interrupt condition is detected by the card, a second interrupting condition will not be tested and stored until at least 10 milliseconds after the preceding one. There is more information about this delay in the advanced programming example at the end of this chapter.

Menu Entries For Model 69776A

Before programming the 69776A Interrupt card, you must specify various parameters in the three page Configuration Menu for the card. The following are descriptions of all the parameters required to configure this function. Most of the parameters on the menus, as well as how to use the menu keys are described in detail in Chapter 3.

```
Name --- Interrupt                Date: 21 Aug 1984
Model 69776A Word Interrupt       Time: 17:37:52
File Name -- Chapter_9
User ID ----
Time-stamp - First Time

                                  Page 1 of 3

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot ..........
Start Bit ..... 0

Timeout ............. 5
```

Menu Page 1 of 3

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ..... 8
HP-IB Address ......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the Interrupt card. It has a legal range from 0 to 7. The default value is 0.

Slot

Identifies the location of the Interrupt card within the frame. It has a legal range from 0 to 15. This parameter **MUST** be filled in; it has no default value.

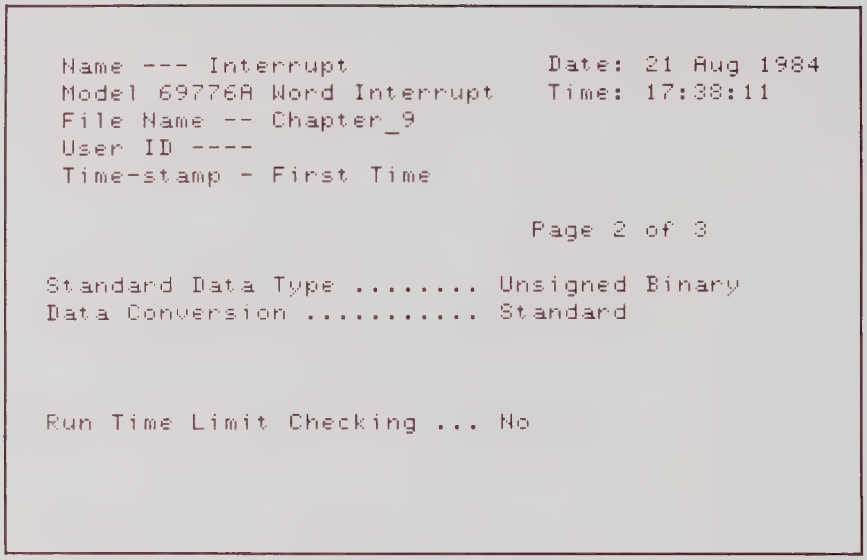
Menu Page 1 of 3 (continued)

Start Bit

Although the interrupt function can perform comparisons on words up to 16 bits in length, you can use words that are smaller. If you choose to use smaller word lengths, however, you must wire the I/O cable such that all of its bits are contiguous, and the unused bits are the Least Significant Bits (LSBs) of the 69776A card's inputs. The value that you enter for Start Bit should be 16 minus the number of bits in your word. For example: a 16 bit word has a Start Bit of 0 and a 12 bit word has a Start Bit value of 4, (16 − 12=4). So, once the value of Start Bit is entered, you don't have to worry about the unused bits when writing your program. For example, if you have entered a Start Bit value of 4, you can program the Reference Word and Mask Word with 12 bit values. The default value for Start Bit is 0.

Timeout

Sets the length of time (in seconds) that will be allowed for the Interrupt card to complete its operation, that is, signal when a true comparison has been detected. If the Wait_for statement does not detect completion within this time period, a Timeout error message (#803) will be generated. Any number between 0.001 and 128,000 may be entered here. The default value is only 5 seconds. The Wait_for statement is used for software synchronization of the hardware function and is discussed in greater detail in Chapter 10.


Name --- Interrupt
Date: 21 Aug 1984
Model 69776A Word Interrupt
Time: 17:38:11
File Name -- Chapter_9
User ID ----
Time-stamp - First Time
Page 2 of 3
Standard Data Type ..... Unsigned Binary
Data Conversion ..... Standard
Run Time Limit Checking ... No

Menu Page 2 of 3

Standard Data Type

Specifies either "Unsigned Binary" or "2's Complement Binary" as the Standard Data Type for the Interrupt card. The default selection is unsigned binary.

Unsigned Binary

The user programs in positive numbers and the input word is an unsigned binary value. The 14752A Output, Preset, Input Read_intr_word and Read_ref_word statements interpret these numbers to be unsigned binary values for the Interrupt card. The range of numbers that can be programmed (for a 16-bit word) is from 0 to 65535. For word lengths less than 16 bits, the range of values are:

0 to $2^{(n-1)}$, where n is the number of bits per word.

2's Complement Binary

The user can program in positive or negative numbers. The 14752A Output, Preset, Input, Read_intr_word, and Read_ref_word statements interpret the negative numbers as 2's complement binary values for the Interrupt card. The range of numbers that can be programmed (for a 16 bit word) is from -32768 to 32767. For word lengths less than 16 bits, the range of values can be calculated from:

$-2^{(n-1)}$ to $[2^{(n-1)}]-1$, where n is the number of bits per word.

Menu Page 2 of 3 (continued)

NOTE

If 2's Complement Binary is selected, hardware modifications on the card are required in order to make the comparison conditions operate correctly. Refer to the 69776A Operating Manual for information on using the 69776A card with 2's complement inputs.

Data Conversion

Specifies how the data is to be scaled for the Output, Preset, Input, Read_intr_word, and Read_ref_word statements. Note that the Write_mask statement is **not** scaled by the Data Conversion option and is not affected by the Data Type that was selected. One of two data conversion options can be selected: Standard and User.

1. "Standard" conversion for this card means that the data values are sent to the card as numbers in the form of unsigned binary or 2's complement values, whichever was selected as the standard data type. Standard (unsigned binary) is the default data conversion for this function.

2. "User" conversion allows data to be specified in terms of the application rather than in terms of the card's Standard Units. As shown below, four additional entries appear on the menu when "User" conversion is selected.

User's Units
P1 P2

Standard
P1 P2

Are values used only when the User Conversion option is selected. These parameters allow you to fill in the ratio of User Units and Standard Units values of the 14752A can convert the user data into information that can be recognized by the card. The default values for P1 are 0; for P2 they are 1.

More About Data Conversion

To illustrate the use of data conversion, suppose a sensor produces a DC voltage which is read by an Analog-to-Digital (A/D) converter. The output from the A/D is connected to the input of the 69776A Interrupt card. The Interrupt card is to be used to signal the computer when the sensor output is greater than a specified value.

The output of the A/D is a number in the 2's complement binary format. The Standard Data Type for the Interrupt card can be chosen to be also the 2's complement binary format. By selecting the Standard Conversion for the card the Reference Word sent to the card will be in the same data format as the output from the A/D.

```
Name --- Interrupt                    Date: 21 Aug 1984
Model 69776A Word Interrupt           Time: 17:38:05
File Name -- CHAPTER_9
User ID ----
Time-stamp - First Time

                                  Page 2 of 3

Standard Data Type ........ Unsigned Binary
Data Conversion ........... Standard


Run Time Limit Checking ... No
```

Menu Page 2 of 3 (continued)

However the 2's complement numbers must be transformed into a number which represents the DC voltage read by the A/D. For example, it is not obvious that the 2's complement number 512 represents a DC voltage of +2.5 volts. By selecting User Conversion, the programmed values for the 69776A Interrupt card's Reference Word value can be specified in terms of the DC voltage of the sensor. The number we send (2.5) now represents a DC voltage level of +2.5 volts.

Selecting User Conversion requires four additional menu entries. The following table shows these entries for the example.

	P1 entry	P2 entry
User Units (Volts)	0	+2.5
Standard Units	0	512

From this example it can be seen that Standard Conversion is used when you want to program a function using numbers it expects. In other words, Standard Units are fundamental units of a function, a 2's complement number in our case above. Standard Conversion is also the fastest way to send data to a function. An Output will execute 20% to 30% faster with the Standard Conversion than with Users Conversion although Users Conversion provides a much easier way to send data to a function.

Run Time Limit Checking If YES is selected, values sent to the Interrupt card by the Output and Preset statements will be checked against High and Low Limits. This optional parameter can be toggled on and off by selecting YES or NO. The default is NO limit checking.

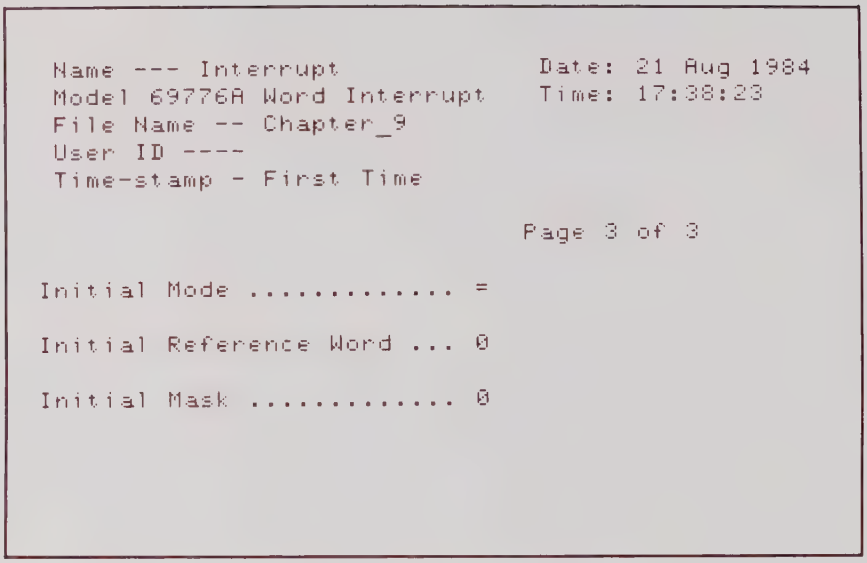
Menu Page 2 of 3 (continued)

High Limit

Is used only when Run Time Limit Checking is ON. Specifies the upper limit for the Reference Word value. The value entered must be greater than the Low Limit specified. An error message (#802) will be generated when a programmed value is greater than this limit. This value should be entered in the same units that are selected for the programming units. This parameter **MUST** be specified; it has no default.

Low Limit

Is used when Run Time Limit Checking is ON. Specifies the lower limit for the Reference Word value. The value entered must be less then the High Limit specified. An error message (#802) will be generated when a programmed output value is less than this limit. This value should be entered in the same units that are selected for the programming units. This parameter **MUST** be specified; it has no default.



Menu Page 3 of 3

The following three menu parameters are used, by the 14752A CAT Programming Package, for the Init and System_init statements. The card's mode, Reference Word, and Mask Word can also be programmed with fundamental programming instructions that are described later in this section.

Initial Mode

Specifies the relationship that must exist between the input data word and the Reference Word to allow the card to detect a true comparison. Four modes are allowed: "equal to", "not equal to", "greater than", and "less than". If you will be using the function in a "Bit Interrupt" configuration you must select the "not equal" mode. The default mode is "equal to".

Initial Reference Word

Specifies the data word used for comparison with the input data word. The values that can be used for the Reference Word are as specified by the Standard Data Type and Data Conversion parameters. The default value is 0.

Initial Mask

Specifies the data word which is used to either include or eliminate specific data bits of the input data word from comparison with the reference word. Data bits are eliminated from the comparison by setting that bit in the Mask Word equal to 0. They are included in the comparison by setting the bit in the Mask Word to a 1. **Note**: this sense of masking bits is opposite to the way the hardware itself works. The range of values for the Mask Word is independent of the Data Type and Data Conversion that is used, but depends on the word length. The range of values for the Mask Word is 0 to 65535 for a 16-bit word length. For word lengths less than 16 bits, the range of values is zero to $2^{(n-1)}$, where n is the number of bits per word. The default value is 0, which will eliminate all the bits from the comparison. For word lengths less than 16 bits, the unused LSBs are automatically excluded from the comparison.

Programming Fundamentals

Only a few fundamental statements are needed to program the Interrupt card. The following is a summary of these statements. For details concerning the syntax requirements for the Name and Value parameters, refer to Chapter 3.

Statement	Description
Init(Name)	Loads the card with the initial data values for Reference Word, Mask Word, Mode, and Timeout that were specified in the Configuration Menu but the card does not test for a comparison until it is started by an External trigger signal, Output statement, or Start statement. Also, interrupts from the function are disabled.
Start(Name1,...,Name8)	Enables the card to perform a comparison. If more than one Name is listed, all the named functions are Started in parallel. The program will continue without waiting for the Interrupt card to detect a true comparison.
Output(Name,Value)	Sends a Reference Word to the function and tells the card to look for a comparison. Note, program execution is **NOT** suspended until the true comparison occurs.
Write_mask(Name,Value)	Sends a Mask Word to the function. The Mask Word allows you to remove certain bits from the comparison. For information about Value, refer to the description of Initial Mask.
Wait_for(Name)	Program execution is suspended until either the function has detected a true comparison, or a timeout occurs. The card must have been previously started with either an Output statement, a Start statement, or an External Trigger signal. Used if you want to wait for the Interrupt card to complete its operation as indicated by a true comparison between the input data word and the Reference Word.
Set_ne(Name)	An input word that is "not equal" to the Reference Word is considered to be a true comparison. This is the mode that you must use to implement the "Bit Interrupt" configuration.
Set_eq(Name)	An input word that is "equal to" the Reference Word is considered to be a true comparison. The "equal to" mode should not be used with the "Bit Interrupt" configuration.
Set_gt(Name)	An input word that is "greater than" the Reference Word will be considered to be a true comparison. The "greater than" mode should not be used with the "Bit Interrupt" configuration.
Set_lt(Name)	An input word that is "less than" the Reference Word will be considered to be a true comparison. The "less than" mode should not be used with the "Bit Interrupt" configuration.

Statement	Description
Output_bit(Name,Bit, Value)	This statement is generally used only in bit interrupt applications. It sets the specified Bit in the Reference word to "Value" (without changing any other bits) and tells the card to look for a comparison using this new value of Reference Word. "Bit" can be 0 through 15 for a 16 bit word length. For word lengths less than 16 bits, the legal range for "Bit" is zero to: 15 minus the value of the Start Bit parameter. Value can be either 0 or 1.
Write_mask_bit(Name, Bit,Value)	Sets the specified Bit in the Mask word to "Value" without changing any other bits. Bit can be 0 through 15 for a 16-bit word length. For word lengths less than 16 bits, the legal range for Bit is zero to: 15 minus the value of the Start Bit parameter. Value can be either 0 or 1. This statement is generally used only in bit interrupt applications.

Example 1. Fluid Level Detection.

The 69776A Interrupt card can be used to detect a certain condition. For example, suppose that the fluid level in a tank is monitored by a sensor which outputs a DC voltage with a range from 0 to 10.240 volts. This DC voltage is converted to a binary number by a 12-bit analog-to-digital converter. The A/D output is connected to the 69776A Interrupt card input and has a value range from 0 to 4095. To signal the computer when the proper fluid level, is reached, the Reference Word is set equal to the value which represents the fluid level. Since we are using a 12-bit input, Start Bit should be set to 4 in the menu.

If a sensor output of 3200 (8.0 volts) represents the critical fluid level the Reference word should be set to 3200. The comparison mode should be set to "greater than". The mask should be set to 4095. These values can be set as the Initial values for the Configuration of the fur ction.

In the program example which follows the 69776A Interrupt card is identified with the Name, "Fluid_level" For simplicity programming of the A/D converter is not included in this example.

Programming Example for Fluid level detection.

```
10 COM/Names/Fluid_level
20 Configure("Ask_me","MYFILE")
30 !
40 ! The Interrupt card is initialized using the
50 ! Reference Word, Mask, Mode, and Start Bit
60 ! specified in the Configure Menu.
70 !
80 Init(Fluid_level)
90 !
100 ! Now we can use a Start statement to start the
110 ! comparison.
120 !
130 Start(Fluid_level)
140 !
150 ! The process starts filling the tank.
160 ! When the tank fills beyond a fixed
170 ! point (the critical level), the program will
180 ! continue, printing a message on the screen.
190 !
200 Wait_for(Fluid_level)
210 PRINT "The Tank Is Almost Full !"
220 !
230 ! The operater can now stop filling the tank.
240 !
250 !
260 END
```

Example 2. Overflow/Underflow Alarm

The previous example, Example # 1, can be expanded so that the computer can detect multiple conditions. After the maximum level has been reached and the tank has started to drain, the computer will be signalled when a specified minimum level has been reached in the tank.

To accomplish this, the reference word is changed (see line 280) to equal the output of the A/D when the tank level has reached the required minimum (1024 in this example). The mode is also changed from "greater than" to "less than" (see line 270).

The program code from the first example is modified as follows to accomplish the additional operation:

Programming Example for Overflow/Underflow Alarm

```
10 COM/Names/Fluid_level
20 Configure("Ask_me","MYFILE")
30 !
40 ! First, we initialize the Interrupt
50 ! card using the Reference Word, Mask
60 ! and Initial Mode as specified in the
70 ! Configuration menus.
80 !
90 Init(Fluid_level)
100 Start(Fluid_level)
110 !
120 ! Now, the process will start filling the tank. The
130 ! program will pause until the tank has filled to
140 ! the desired level by using the Wait_for statement.
150 ! A PRINT statement notifies the operator.
160 !
170 Wait_for(Fluid_level)
180 PRINT "Tank Is Almost Full."
190 !
200 ! At this point, the operator can
210 ! start draining the tank. The
220 ! Interrupt card's Reference Word will be
230 ! changed using the Output statement and the
240 ! Mode will be changed from "Greater Than' to
250 ! "Less Than".
260 !
270 Set_lt(Fluid_level)
280 Output(Fluid_level,1024)
290 !
300 ! The program will now continue when the
310 ! level in the tank reaches the minimum (1024).
320 !
330 Wait_for(Fluid_level)
340 PRINT "The Tank Is Almost Empty."
350 END
```

Advanced Programming

There are 14752A statements that allow special purpose programming of the Interrupt card. They are considered advanced programming statements because you need a good understanding of the Multiprogrammer I/O cards and interrupt system. They allow features such checking status, generating interrupts, and reading miscellaneous data. Most of these features are explained in greater detail in Chapter 10. The following is a list of the advanced programming statements which can be used with the 69776A Interrupt card.

Statement	Description
Preset(Name,Value)	Sends a Reference Word to the function. Used as a set-up operation for the Start statement and External Trigger. Finishes quickly because it does not have to wait for a response from the card.
Input(Name,Value)	Reads the data word that is currently at the input (card edge) of the Interrupt card. Formats the data and returns the data into the variable, Value.
Read_intr_word(Name, Value)	Reads the data word which has caused the comparison to become true. This data word has been stored on the card and is not to be confused with the current input word which is read by the Input statement. Formats the data and returns the data into the variable, Value.
Check_done(Name,Flag)	Like Wait_for, this statement inquires about the status of the function's operation. However, it will not suspend program execution. Flag is set to a one if the comparison between the input word and the reference word is true. Once a value of one is returned, the Check_done statement will subsequently return a zero until the function again detects a true comparison.
Enable_intr(Name,Vector)	Allows the 69776A Interrupt card to signal the computer via the interrupt system when a true comparison exists between the reference word and the input data word. Must be used in conjunction with the interrupt statements a described in Chapter 10 of this manual. **Note:** if this statement is used, the Init statement will cause the interrupts to become disabled as if a Disable_intr statement was executed. When the 69776A card interrupts, the User_handler subprogram is passed a variable called "Reason" as explained in Chapter 10. Reason for the 69776A card is defined by the Boolean expression: [(interrupting word) xor (Reference Word)] and (Mask Word) Or, more simply, a 16-bit word with only the interrupting bits set to 1.
Disable_intr(Name)	Inhibits the Interrupt card from signalling the computer when the comparison condition becomes true. Meaningful only when the Enable_intr statement has already been programmed.

Statement	Description
Set__timeout(Name,Value)	Changes the value of timeout from the time specified in the menu to Value. The legal range for Value is from 0.001 to 128,000 seconds.
Read__ref__word(Name, Value)	Reads the Value of the Reference word. Formats the data and returns the data into the variable, Value.
Preset__bit(Name,Bit,Value)	Sets the specified Bit in the Reference Word to Value but dones not tell the card to look for a comparison. Refer to the Output__bit statement for a description of Bit and Value. Used as a "set-up" operation for a Start statement or External Trigger signal.
Input__bit(Name,Bit,Value)	Reads the Value of the specified Bit of the data word that is currently at the input of the Interrupt card. Refer to the Output__bit statement for a description of Bit and Value.

Advanced Programming Example Using The Word Interrupt Configuration

In Example #2, above, a single Interrupt card was used as both the overflow and the underflow detection for a fluid tank. The following example illustrates how to use the 69776A Interrupt card in conjunction with the computer's interrupt system as an alarm monitor. For this example two 69776A Interrupt cards will be used. The output of the fluid level sensor is connected to both of the Interrupt cards. One of the Interrupt cards is the overflow alarm and the other is the underflow alarm. Both cards will be enabled to check for a comparison by using the Start statement. The interrupt system will be used to signal the computer so that the program is not paused waiting for an alarm condition to occur. The Enable__intr statement is used to allow the cards to signal the computer. This statement also allows an "interrupt vector" to be specified so that the interrupt handling software can easily determine which of the two cards has interrupted, and thus, which condition has occurred. The interrupt system and the use of interrupt vectors is extensively discussed in Chapter 10 of this manual.

The two Interrupt cards are identified with the Names, "Overflow" and "Underflow". The Initial Mode for "Overflow" is set to "greater than" and the Initial Mode for "Underflow' is set to "less than".

Advanced Programming Example for Under/Overflow Alarm

```
10 COM/Names/Overflow,Underflow
20 Configure("Ask_me","MYFILE")
30 !
40 ! The two Interrupt cards are initialized
50 ! using the initial conditions specified
60 ! in the Configuration menus.
70 !
80 Init(Overflow)
90 Init(Underflow)
100 !
110 ! First, the interrupts for the two cards
120 ! are enabled. 1 and 2 are used as Vector
130 ! numbers. Refer to Chapter 10 for a more
140 ! complete explaination of the use of
150 ! the Intr_handler subprogram and vector numbers.
160 !
170 ON INTR 29,3 CALL Intr_handler
180 Enable_sc_intr(29)
190 Enable_intr (Overflow,1)
200 Enable_intr (Underflow,2)
210 ! Now, the two Interrupt cards can be enabled
220 ! to monitor the status of the fluid sensor
230 ! in the tank.
240 !
250 Start(Overflow,Underflow)
260 !
270 ! Now the program can continue with the testing
280 ! that is to be performed (in this example
290 ! we are just looping on a single line). When an
300 ! interrupt occurs, the subprogram "User_handler"
310 ! will be executed; refer to Chapter 10.
320 !
330 GOTO 330
340 END
```

NOTE: Any Init(Name) statement after line 200 will disable interrupts from the 69776A. In such a case additional Enable_intr statements may be required.

Advanced Programming Example Using The Bit Interrupt Configuration

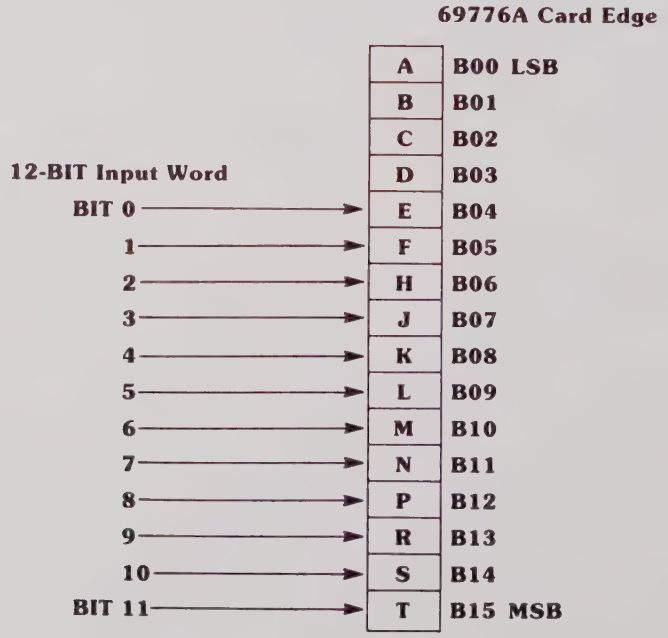
Although the 69776A Interrupt Card was primarily designed to compare two multi-bit digital words (which represent numeric values) with each other, it is possible, if you understand some limitations, to have the individual bits represent independent interrupt signals. In this configuration, the Reference Word represents the quiescent state of the various input bits; input bits which you do not want to cause interrupts can be masked out. The card is set to the "not equal" mode.

It is important to remember, however, that the bits are not truly independent. The card has only a single interrupt line to the computer to handle all the input bits. Once one bit changes and an interrupt signal is sent to the computer, other input bits (your "interrupts") can pulse without ever being detected, if they happen before you tell the card to look for another interrupt by means of an "Output", "Output_bit", or "Start" instruction in your interrupt service routine "User_handler" as explained in Chapter 10. It takes 7 milliseconds (minimum) from the time an interrupt is requested by the 69776A Interrupt Card until the statements in "User handler" begin executing; once "User_handler" completes, there is another 3 millisecond minimum delay for the interrupt service routine to finish up.

What this means for the 69776A Interrupt Card is that there is a 7 millisecond minimum delay from the time one "interrupt" is detected until the card hardware can begin to look for another; and the minimum time period from the start of servicing one "interrupt" to the start of servicing a second is 10 milliseconds plus the time you spend in "User_handler".

With these limitations in mind, we can proceed to the following example. In this example, a 12-bit digital input word has been specified by setting the Start Bit parameter in the menu to 4. The 12-bit word is wired to the 69776A card edge connector as shown in the figure below. Three of the bits will be allowed to interrupt, if they deviate from their normal state as defined below:

Bit 0 (wired to input line 4): normally 0
Bit 8 (wired to input line 12): normally 0
Bit 11 (wired to input line 15): normally 1



Advanced Programming Example Using the Bit Interrupt Configuration

```
10 COM /Names/ Alarm
20 Configure ("Ask_me","MYFILE")
30 !
40 ! In the menu, an Initial Mask of 0 (all bits masked out)
50 ! and an Initial Reference Word of 0 were set.
60 ! The Initial Mode was set to "not equal",
70 ! as required when using the input lines
80 ! as individual bits.
90 !
100 Init(Alarm)
110 !
120 ! The normal state of the 3 input bits are defined:
130 !
140 Preset_bit(Alarm,0,0)
150 Preset_bit(Alarm,8,0)
160 Preset_bit(Alarm,11,1)
170 !
180 ! Now, the bits are unmasked:
190 !
200 Write_mask_bit(Alarm,0,1)
210 Write_mask_bit(Alarm,8,1)
220 Write_mask_bit(Alarm,11,1)
230 !
240 ! The card is enabled to interrupt,
250 ! and begins looking for a non-equal
260 ! input word. Refer to Chapter 10
270 ! for a more thorough treatment of
280 ! interrupts.
290 !
300 ON INTR 29,3 CALL Intr_handler
310 Enable_sc_intr(29)
320 Enable_intr(Alarm,1) ! Note that 1 is the value of Vector.
330 Start(Alarm)
340 !
350 ! The program will continue until
360 ! an interrupt occurs.
370 !
380 GOTO 380
390 END
```

It is advantageous for your hardware to be able to sustain an interrupting signal until the condition that caused it is serviced by your "User_handler" subprogram. In this way, you can use the Input statement during User_handler to see if any interrupt requests other than the one which caused the interrupt need to be serviced as well. This speeds up interrupt response time, and can insure that interrupt requests from your external hardware will not be missed.

69790B Memory Card

The 69790B Memory card consists of two separate printed circuit boards connected by a ribbon cable. Circuit board #1 contains a 4096 word by 16-bit block of RAM memory plus handshake control circuits. Circuit board #2 has four counter/registers which control addressing and read/write access to the RAM memory. A complete hardware description can be found in the 69790B Operating Manual (HP P/N 69790-90003).

Other Multiprogrammer cards or external devices can read data from and write data to the Memory card via its external edge connector. The computer writes and reads data through the Memory card's connection to the Multiprogrammer. Since there is a different data path used by an external device than by the computer to communicate to the Memory card, it is possible to simultaneouely write data to and read data from the Memory card. This powerful feature permits operations such as continuous data acquisition to be easily performed. There is detailed information on how to do continuous data acquisition in Chapter 10.

There are four operating modes or ways to read and write data on the Memory card. They are: the FIFO (First In First Out) Input, the Recirculate Input, the FIFO Output, and the Recirculate Output modes.

The FIFO Input mode allows the Memory card to act like a simple data buffer. That is, the data written to the card by an external device is stored in sequential memory locations. The card will continue to accept data until data has been stored in the last location on the card. This is referred to as a "Memory Full" condition. After the Memory Full condition occurs, subsequent data inputs will be ignored so as not to overwrite previous data on the Memory card. The memory address where the data is to be written is referred to as the "Write Pointer". The Write Pointer is automatically updated after each data item is written into the memory card. The write pointer is programmable so that data can be written into the Memory card starting anywhere within the 4096 locations on the Memory card.

The FIFO input mode is also used to read data from the Memory card into the computer. The data is read from the card in the same order that it was written into the card. The memory address, where the data is read from, is referred to as the "Read Pointer". The Read Pointer is automatically updated after each data item is read from the card. The Read Pointer is also programmable so that data can be read from the Memory card starting anywhere within the 4096 locations of the card.

In the Recireulate Input mode, the Memory card acts like a circular buffer. Data from an external device will be stored in sequential locations on the Memory card until the last location has been written to. The next data item will then be stored on the Memory card starting at the first memory location, overwriting the previously stored data item. The Memory card will thus contain the most current data. As with the FIFO Input mode, the Write Pointer is the address of the location to be written into and is automatically updated after each data word is written.

The Recirculate Input mode can also be used to read data from the Memory card into the computer. Data is read from sequential memory locations as determined by the Read Pointer. Once the data in the last location is read, the Read Pointer will be automatically reset to the first memory location.

The FIFO Output mode is used to send data to the Memory card from the computer. Each data item is placed in sequential memory locations until the Memory Full condition occurs. If more data items are sent to the card, the program will wait until either some data is taken from the card by the external

device, or a Timeout error occurs. The memory address where each data item is placed is determined by the Write Pointer.

The FIFO Output mode also allows the data stored in sequential memory locations on the Memory card to be sent to an external device. The data sent to the external device is controlled by the Read Pointer.

The Recirculating Output mode can be used to send data to the Memory card from the computer. Data is sent to sequential memory addresses on the card, starting at the address of the Write Pointer. Note, the Reference Count is ignored when sending data to the Memory card while in Recirculating output mode. Therefore, You can fill the entire card with data and then truncate the amount of data that is sent to the external device by using the Reference Count.

The Recirculate Output mode allows data, previously stored on the Memory card, to be output repetitively to an external device. Starting with the location pointed to by the Read Pointer, data items in sequential memory locations are sent to the external device. The Read Pointer is automatically incremented after each data word is read by the external device. The amount of output data can be controlled by programming the "Reference Count" to a value equal to the address of the last data item you want to output. When the Read Pointer address equals the Reference Count, the Read Pointer will be reset to zero and data will continue to be output starting again from the first memory location. The output process will continue in this fashion, allowing you to generate continuous streams of data.

Menu Entries For Model 69790B

Before programming the 69790B Memory card, you must specify various parameters in the Configuration Menu for the card. Following are descriptions of all the parameters required to configure this function. Many of the menu parameters, as well as how to use the menu keys, are described in detail in Chapter 3.

```
Name --- Memory                    Date: 21 Aug 1984
Model 69790B 4K Word Memory        Time: 17:39:31
File Name -- Chapter_9
User ID ----
Time-stamp - First Time

                                 Page 1 of 3
Multiprogrammer type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot ..........


Timeout ....... 5
```

Menu Page 1 of 3

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ...... 8
HP-IB Address .......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the Memory card set. It has a legal range from 0 to 7. The default value is 0.

Slot

Identifies the location of the Memory card (circuit board # 1) within the frame. It has a legal range from 0 to 14. This parameter **MUST** be filled in; it has no default value. Circuit Board # 2 **MUST** be in the next higher slot.

Menu Page 1 of 3 (continued)

Timeout

Specifies the time period in seconds that will be allowed for various operations, as listed below:

1. **Output_rblock or Output_iblock statements (FIFO output mode only).** The time allowed for the specified number of words of the data in the array to be transferred from the computer into the Memory card. Program execution is suspended until the transfer is completed. If the data is not transferred within the time period specified, an error message (#803) is generated.
2. **Input_rblock or Input_iblock statements (FIFO input mode only).** The time allowed for the specified number of words of data to be read into the computer array from the Memory card. Program execution is suspended until the transfer is completed. If the data is not transferred within the time period specified, an error message (#803) is generated.
3. **Wait_for statement (FIFO output mode only).** The time allowed for the memory card to "empty out"; that is, for the number of words remaining to be read from the card by an external device to become equal to, or less than, the value of the Reference Count. If a Wait_for statement does not detect this condition within the time period specified, error message (#803) is generated.
4. **Wait_for statement (FIFO input mode only).** The time allowed for the memory card to "fill up"; that is, for the number of words written into the card by an external device, but not yet read by the computer, to exceed the value of the Reference Count. If a Wait_for statement does not detect this condition within the time period specified, error message (#803) is generated.

The timeout period entered can be any number from 0.001 and 128,000. The default is 5 seconds. Note that Timeout does not apply to a 69790B Memory card that is in either the recirculate input or recirculate output mode.

```
Name --- Memory                    Date: 21 Aug 1984
Model 69790B 4K Word Memory        Time: 17:39:39
File Name -- Chapter_9
User ID ----
Time-stamp - First Time

                                 Page 2 of 3

Standard Data Type ........ Unsigned Binary
Data Conversion ........... Standard


Run Time Limit Checking ... No
```

Menu Page 2 of 3

Standard Data Type — Specifies either "Unsigned Binary" or "2's Complement Binary" as the Data Type. The default selection is Unsigned Binary.

Unsigned Binary — The user reads or writes data in positive numbers only. The 14752A Output_rblock, and Input_rblock statements interpret these numbers to be unsigned binary values for the Memory card. The range of valid numbers is from 0 to 65535.

2's Complement Binary — The user can read or write data in positive or negative numbers. The 14752A Input_rblock and Output_rblock statements interpret these negative numbers as 2's complement binary values for the Memory card. The range of numbers than can be read or programmed is from −32768 to +32767.

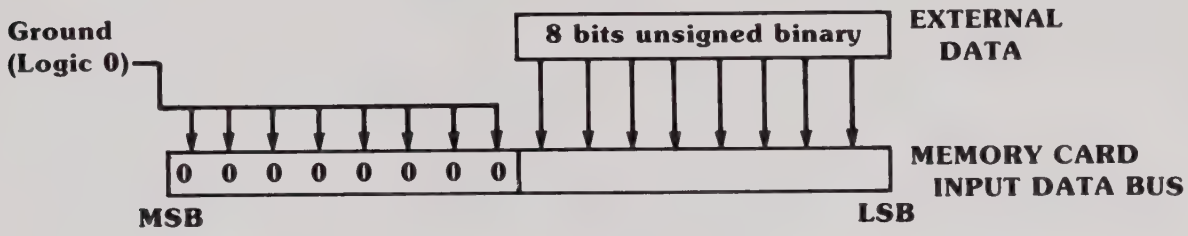
Using the Memory Card for Word Lengths Less Than 16 Bits

Any bits of the memory card's external data input bus which are left unconnected will assume a value of 1 when words are written to memory by an external device in the FIFO input or Recirculate input modes of operation.

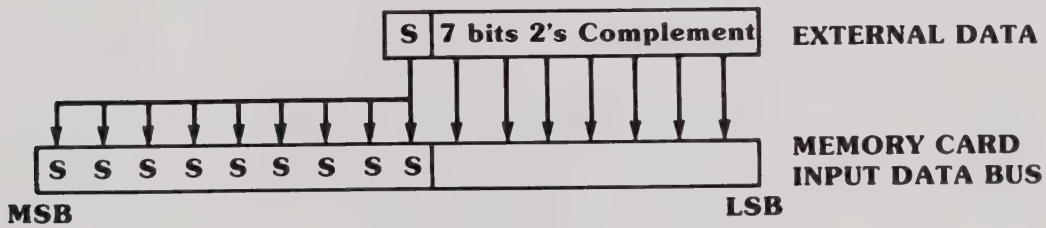
You can make up your I/O cables for the memory function in a way that will avoid this undesirable situation. The cable wiring depends on the Standard Data Type you will be using.

Menu Page 2 of 3 (continued)

Unsigned Binary: the external data to be stored is an 8-bit unsigned binary value.



2's Complement: the external data to be stored is an 8-bit 2's complement value (sign bit + 7 data bits)



Data Conversion

Specifies how the data is to be scaled for the Input__rblock and the Output__rblock statements. One of two data conversion options can be selected: Standard and User. Note that for either an Output__iblock or an Input__iblock statement, the Data Conversion parameter does not apply; refer to the statement descriptions under "Advanced Programming" for information about the Output__iblock and Input__iblock statements.

1. "Standard" conversion for this card means that the data values in an Output__rblock or Input__rblock statement are in the form of unsigned binary or 2's complement values, whichever was selected as the standard data type. Standard (unsigned binary) is the default data conversion for this function.

2. "User" allows data to be specified in terms of the application rather than in terms of the card's Standard Units. As shown below, four additional entries appear on the menu when "User" conversion is selected.

```
Name --- Memory                    Date: 21 Aug 1984
Model 69790B 4K Word Memory        Time: 17:39:51
File Name -- CHAPTER_9
User ID ----
Time-stamp - First Time

                                Page 2 of 3

Standard Data Type ........ Unsigned Binary
Data Conversion ........... Standard


Run Time Limit Checking ... No
```

Menu Page 2 of 3 (continued)

User's
P1 P2

Standard
P1 P2

These values appear on the menu only when the User Conversion option is selected. These parameters allow you to fill in the ratio of User's to Standard values so the 14752A can scale the data as it is sent to, or read from, the Memory card. The default values for P1 are 0; for P2 they are 1.

More About Data Conversion

To illustrate the use of data conversion, assume we are using the Memory card to supply data to a Digital-to-Analog converter. The output of this converter is programmable from 0 to +20.480 volts for a digital input value from 0 to 4096. If "Standard" units were used, the data sent to the Memory card would be in the form of REAL numbers from 0 to 4096. We would like, however, to send data to the Memory card in terms of the output voltage of the D/A. Therefore, "User" conversions is selected in the menu. This will allow the program data to be in the form of output voltage and the 14752A will convert this data into the data format required by the D/A.

When "User" conversion is selected in the menu, four other parameters must be also be specified. These parameters give the 14752A all the data it needs to scale the user's units (volts) to "Standard" units. To perform the conversion, it is necessary to specify two points, for instance the minimum output and the maximum output. These two points will specify the two parameters for the 14752A. In this example, the minimum output of 0 volts equals a data value of 0 and the maximum output of 20.480 volts equals a data value of 4096. The range of values would be as follows:

	minumum output (P1)	maximum output (P2)
User's Units	0 volts	20.480 volts
Standard Units	0	4096

Menu Page 2 of 3 (continued)

From this table we can see that: User's P1 = 0
User's P2 = 20.480
Standard P1 = 0
Standard P2 = 20.480

Even though User's Unit may be more convenient, it takes the 14752A 20% to 30% longer to fill the card with this data than if Standard units were specified.

Run Time Limit Checking — If YES is selected, values sent to the Memory card using the Output_rblock statement will be checked against the specified High and Low Limits. Values which are read from the Memory card by the computer, or values that are sent to it by an "Output_iblock" statement, are not checked against either the High Limit or the Low Limit. This optional parameter can be toggled on or off, by selecting YES or NO. The default is NO limit checking. When Run Time Limit checking is selected, the following two parameters **MUST** be filled in; they have no default.

High Limit — Is used only when the Run Time Limit Checking is ON. Specifies the upper limit for the data values specified in the Output_rblock statement. The value entered must be greater than the Low Limit specified. An error message (#802) will be generated when a programmed output value is greater than this limit. This value should be entered in the same units that are selected for the programming units.

Low Limit — Is used when the Run Time Limit Checking is ON. Specifies the lower limit for the data values specified in the Output_rblock statement. The value entered must be less than the High Limit specified. An error message (#802) will be generated when a programmed output value is less than this limit. The value should be entered in the same units that are selected for the programming units.

```
Name --- Memory                    Date: 21 Aug 1984
Model 69790B 4K Word Memory        Time: 17:39:59
File Name -- CHAPTER_9
User ID ----
Time-stamp - First Time

                                Page 3 of 3

Initial Write Pointer ..... 0
Initial Read Pointer ..... 0
Initial Reference Count ..... 0


Initial Mode ..... FIFO In
Initial Lockout ..... On
```

Menu Page 3 of 3

Initial Write Pointer

Specifies the location on the Memory card where the first data value will be written to. The legal range for this parameter is from 0 to 4095. The default value is 0.

Initial Read Pointer

Specifies the location on the Memory card from which the first data value will be read. The legal range for this parameter is from 0 to 4095 with a default value of 0.

Initial Reference Count

Specifies the value programmed into the Reference Register on the Memory card when an Init or System_init statement is programmed. The function of the Reference Count depends upon the mode of operation that is in effect:

1. In the FIFO Input mode the Reference Count sets the number of words that will be stored into the Memory card by the external device before it will indicate that memory is "almost full".

2. In the FIFO Output mode the Reference Count sets the number of words that remain in the card before it will indicate that it is "almost empty".

3. In the Recirculate Output mode the Reference Count truncates, i.e. sets a limit on, the size of the memory being read from the card by an external device. Note, the Reference Count has NO effect when sending data to the card from the computer, while in the Recirculate Output mode.

Menu Page 3 of 3 (continued)

4. In Recirculate Input mode the Reference Count is meaningless.

   Indication of "almost empty" or "almost full" can be checked with the statements Check__done, Wait__for, or Enable__intr. All of these statements are described in the Advanced Programming section later in this chapter. The "almost full" and "almost empty" conditions are also indicated by the setting of a signal called EOP that is located at the edge connector (J2-n) of Memory card No. 1.

Initial Mode

Either the FIFO Input, FIFO Output, Recirculate Input, or Recirculate Output mode can be selected. The default mode is FIFO Input.

Initial Lockout

The Memory card can be inhibited from either inputting or outputting data from/to an external device by turning the External Lockout ON. Lockout is used when you want to transfer data in or out of the computer without the interruption of an external device. The lockout can be either turned ON or OFF. The default is Lockout ON.

Programming Fundamentals

Only a few fundamental statements are needed to program the Memory card. The following is a summary of these statements. For details concerning the syntax requirements for the Name parameter, refer to Chapter 3.

Statement	Description
Init(Name)	Loads the card with the initial values for mode, external lockout, Reference Count, Read Pointer, Write Pointer, and Timeout as specified in the Configuration Menu. This instruction also disables interrupts from the Memory card if they have been enabled with the advanced programming instruction, Enable_intr.
Input_rblock(Name, Array(*)[,Count [,Index]])	Reads a block of data from the Memory card, converts the data according to the Standard Data type and Data Conversion specified, and returns the data in the REAL array called 'Array'. The card must be in either the FIFO input or Recirculate input mode; if it is not, an error message (code 825) will be generated. The number of data words read from the Memory card is equal to the size of the data array. That is, if the data array is dimensioned for 1000 then 1000 data words will be returned. Optionally, if the number of data words to be read is less than the size of the data array then a parameter, referred to as "Count", can be specified for the required number of data words. If "Count" is used, you can also use the optional parameter, "Index". Index can be used with the Count parameter to specify the position in the array where the first input data word will be placed; subsequent data words are placed in the successive array positions that follow. The data array **MUST** be previously DIMensioned or ALLOCATEd in the user's program, otherwise a BASIC system error (#13) will be generated.
Output_rblock(Name, Array(*)[,Count[,Index]])	Converts a block of REAL data according to the Standard Data Type and Data Conversion that was selected in the menus and then sends the block to the Memory card. The card must be in either the FIFO output or Recirculate output mode; if it is not, an error message (code 825) will be generated. The number of words written to the Memory card is equal to the size of the data array called "Array". That is, if the data array is dimensioned for 2048 words, then 2048 words will be written to the Memory card. Optionally, if the number of data words to be sent is less than the size of the data array than the parameter,"Count", can be specified for the required number of data words that will be sent to the Memory card. The optional parameter, "Index" can be used with the Count parameter to specify the array position of the first data word that will be sent. The data array **MUST** be previously DIMensioned or ALLOCATEd in the user's program, otherwise a BASIC system error (#13) will be generated.

The operating mode of the Memory can be changed from that selected in the Configuration phase by using one of the following four statements.

Statement	Description
Set_fifo_in(Name)	Changes the Memory card's operating mode to the FIFO Input mode.
Set_fifo_out(Name)	Changes the Memory card's operating mode to the FIFO Output mode.
Set_recirc_in(Name)	Sets the Memory card's operating mode to the Recirculate Input mode.
Set_recirc_out(Name)	Sets the Memory card's operating mode to the Recirculate Output mode.

Example #1 FIFO Input

One of the ways in which the 69790B Memory card can be used is a simple data buffer. Suppose that we wish to store a number of data words generated by an external device and read them back into the computer. No data conversion will be required in this example; only 16-bit words will be transferred from the external device through the Memory card to the computer.

Programming Example for a Simple Data Input

```
10 COM/Names/Fifo_in
20 Configure("Ask_me","MYFILE")
30 REAL R_buffer(1024)
40 !
50 ! The Memory card is initialized with the Read
60 ! and Write Pointers set to zero, Initial Mode
70 ! set to FIFO Input and the Initial Lockout OFF.
80 !
90 Init(Fifo_in)
100 !
110 ! The Memory card can now accept data from the
120 ! external device. Assume that the device
130 ! requires 2 seconds to generate 1024 data words.
140 !
150 WAIT 2 ! wait for data
160 !
170 ! The data has been stored in the Memory card
180 ! by the external device and can now be read back into
190 ! the computer using the Input_rblock statement
200 !
210 Input_rblock(Fifo_in,R_buffer(*))
220 !
230 END
```

Example 2 FIFO Output

The 69790B Memory card is bi-directional, that is, it is not only an input device but also an output device. The previous example showed how to read data from the Memory card into the computer. This example will illustrate how to output a block of REAL data to the Memory card.

In this example assume that a data array has been dimensioned and that this array is used by different routines in the program. In other words, it is a general purpose array. Only the first 2048 data words in this array are to be output to the Memory card. This requires the use of an optional parameter, called "Readings", in the Output_rblock statement.

Programming Example for a Simple Data Output

```
10 COM/Names/Fifo_out
20 Configure("Ask_me","MYFILE")
30 !
40 ! General purpose array, R_buffer, is
50 ! dimensioned and the optional parameter,
60 ! Readings is specified as a REAL variable
70 !
80 REAL R_buffer(5000)
90 REAL Readings
100 !
110 ! The Memory card is initialized with the Mode
120 ! set to FIFO Output, Initial Lockout OFF, and the
130 ! Read and Write Pointers set to zero. Since the data
140 ! will be in User Units or REAL numbers, data conversion
150 ! must be specified.
160 !
170 Init(Fifo_out)
180 !
190 ! Assume that the data to be output to the Memory
200 ! card had been stored in the first 2048 positions of the
210 ! array, R_buffer. The variable, Readings, is set to the
220 ! number of words which are to be output to the card.
230 !
240 Readings=2048
250 !
260 ! The first 2048 data words contained in the array are
270 ! now output to the Memory card.
280 !
290 Output_rblock(Fifo_out,R_buffer(*),Readings)
300 !
310 ! Now let's use Index to send the next 100 values of the
320 ! array, starting with array element 2049:
330 !
340 Output_rblock(Fifo_out,R_buffer(*),100,2049)
350 END
```

Advanced Programming

There are 14752A statements which allow special purpose programming of the Memory function. They are considered advanced programming statements because you need a good understanding of the Multiprogrammer I/O cards and system to put together a meaningful program. In general, these statements are used only when precise control of the card's data flow is a concern. They allow features such as: interrupts, reading from and writing to the card registers, and checking card status. Most of these features are explained in greater detail in Chapter 10. The following is a list of the advanced programming statements which can be used for the 69790B Memory function.

The following two statements, Input_iblock and Output_iblock, do not format the data according to the data conversion option selected; that is, the data is returned in exactly the form that it is stored on the Memory card. These two statements can be used to speed up the data transfer rate between the Memory card and the computer. There is a approximately a 2X speed improvement using these statements.

Statement	Description
Input_iblock(Name, Iarray(•)[,Count[,Index]])	Similar to the Input_rblock statement. Reads a block of 16-bit 2's complement data from the Memory card and returns this data in the INTEGER array called "Iarray". The card must be in either the FIFO input or the Recirculate input mode; if not, an error message (code 825) will be generated. The number of data words read from the Memory card is equal to the size of the data array, that is, if the data array is dimensioned for 1000 then 1000 data words will be returned. The optional parameter, Count, specifies the number of data words that will be read from the Memory card. The value of Count must be less than, or equal to the size of the data array. The optional parameter, Index, can be used with the Count parameter to specify the array location where the first data word will be placed. The data array **MUST** be previously DIMensioned or ALLOCATEd in the user's program otherwise a BASIC system error (# 13) will be generated.
Output_iblock(Name, Iarray(•)[,Count[,Index]])	Similar to the Output_rblock statement. Writes a block of 16-bit 2's complement INTEGER data to the Memory card. The card must be in either the FIFO outout mode or the Recirculate output mode; if it is not, an error message (code 825) is generated. The optional parameter, Count, can be used to specify the number of data words to be sent to the card. If Count is not specified, the number of words written to the Memory card is equal to the size of the data array. That is, if the data array is dimensioned for 2000 words then the first 2000 words of the array will be written to the card. The optional parameter, Index, can be used with Count to specify the array location of the first data word that will be sent. The data array **MUST** be previously DIMensioned or ALLOCATEd in the user's program, otherwise a BASIC system error (# 13) will be generated.

Statement	Description
Translate(Name,Iarray(*), User(*))	Used to process data according to the data conversion specified on page 2 of the 69790B configuration menu. This statement enables you to separate the data conversion process from the actual data transfer. If the memory is in an input mode, the INTEGER data in "Iarray" is converted to REAL numbers and placed into the "User" array. If the memory is in an output mode, the REAL data in the User array is converted to INTEGER data and placed into Iarray. Arrays User and Iarray **MUST** be previously DIMensioned or ALLOCATEd in the user's program, otherwise a BASIC system error (#13) will be generated.
Check_done(Name,Flag)	Inquires about the status of the Memory card's operation. Flag returns a value of one if: 1. In the FIFO Input mode, the number of words written to the Memory card by an external device, but not read by the computer, is greater than the value stored in the Reference Count This is called the "almost full" condition. 2. In the FIFO Output mode, the number of words to be read from the Memory card, by an external device, is equal to, or less than, the value stored in the Reference Count This is called the "almost empty" condition. If neither of these conditions occur, the variable Flag, is set to zero. Also, if the Memory card is in either the Recirculate Input or the Recirculate Output mode, Flag always returns a value of zero.
Wait_for(Name)	Like the Check_done statement, Wait_for also checks the cards status in the FIFO modes; however, program execution is suspended until one of the two conditions, described above, occur. If neither of the two conditions occur before the Timeout value has elapsed, an error message (#803) is generated. This statement is used if you want to wait for the Memory card to indicate an "almost full" or "almost empty" condition while in either the FIFO input or FIFO output mode, respectively. The Wait_for statement is only useful in either of the two FIFO modes; an error message (#803) will be generated if the Memory card is in one of the Recirculate modes and a Wait_for statement referencing the Memory card is encountered in the program.
Read_rpointer(Name, Value)	Returns the address of the next memory location to be read from. The address is placed in the variable, Value.
Read_wpointer(Name, Value)	Returns the address of the next memory location to be written into. The address is placed in variable, Value.

Statement	Description
Read_ref_count(Name, Value)	Returns the value stored in the Reference Register of the Memory card. The value is placed in variable, Value.
Read_dif_count(Name, Value)	Returns the difference between the Memory card's Read and Write Pointers. The difference is placed in variable, Value.
Write_rpointer(Name, Value)	Sets the Memory card's Read Pointer to the address in the variable, Value, and adjusts the value of the Difference Count. The range of legal addresses is 0 through 4095.
Write_wpointer(Name, Value)	Sets the Memory card's Write Pointer to the address in the variable, Value and adjusts the value of the Difference Count. The range of legal addresses is 0 through 4095.
Write_ref_count(Name, Value)	Sets the Memory card's Reference Count register equal to the number contained in the variable, Value. The range of legal values is 0 through 4095.
Write_dif_count(Name, Value)	Sets the Memory card's Difference Count register equal to the number in the variable, Value. Note, the Difference Count is adjusted (to reflect the true difference between the write and read pointers) every time that either the Read or the Write pointer is changed. So, the value you write to the Difference Count will be changed if either the Read or Write pointer is written to. The range of values is 0 through 4095.
Enable_lockout(Name)	Prevents data from being written into, or read out of, the Memory card by an external device. Used to prevent an external device from changing any data or pointers.
Disable_lockout(Name)	Allows the Memory card to respond to write/read requests from an external device. The Disable_lockout statement does this by enabling the Memory card's edge connector handshake lines.
Enable_intr(Name,Vector)	Allows the 69790B Memory card to signal the computer via the interrupt system when either the "almost empty" or "almost full" condition occurs. The Memory card can be programmed to interrupt the computer only when it is in either the FIFO input mode or the FIFO

Statement	Description
	output mode. The conditions under which the Memory card will interrupt the computer are: 1. In the FIFO Input mode, the computer will be signalled when the number of words written to the Memory card by the external device but not read from it by the computer is greater than the value stored in the Reference Count (the "almost full" condition). For example, suppose that when 2500 words are stored in the Memory card by an external device the computer should be informed to read them. The Reference word is set equal to the number of data items required or 2500. When data item # 2500 is read into the Memory card the computer is informed. The computer can then read the entire 2500 word data block. 2. In the FIFO Output mode, the computer will be signalled when the number of words yet to be read from the Memory card by an external device, is equal to or less than the value stored in the Reference Count (the "almost empty" condition). For example, suppose several large blocks of data are to be transferred from the computer to an external device. The Reference Word is set to a value that will allow the computer to send a block of data to the Memory card before the external device can read the last data item from the Memory card. When the number of data items to be read from the Memory card becomes equal to the Reference Word the computer is signalled. The computer will then send a new block of data to the Memory card. The process will continue until all the blocks of data in the computer have been sent to the external device. The Enable_intr statement must be used in conjunction with the interrupt statements as described in Chapter 10.
Disable_intr(Name)	Inhibits the Memory card from signalling the computer when it's almost full or almost empty. Meaningful only when the Enable_intr statement has already been programmed. Note, the Init statement will also disable interrupts.
Set_timeout(Name,Value)	Changes the value of timeout from the time specified in the menu to the new Value. The range of legal values is 0.001 and 128,000 seconds.

Advanced Programming Example

To illustrate some of the advanced programming features, Example # 1 of the Fundamental Programming section, has been modified for the example below. The Write_ref_count and the Wait_for statements are used instead of the BASIC WAIT statement. In this way, program execution will be paused only for as long as is needed for the external device to write the required amount of data onto the Memory card. Also, the program will wait any additional time that the the external device may require. A Timeout value should be set during the Configuration phase to allow enough time for the external device to load the data into the card.

Advanced Programming Example for Data Input

```
10 COM/Names/Fifo_in
20 Configure("Ask_me","MYFILE")
30 REAL R_buffer(1024)
40 !
50 ! The Memory card is initialized with the Read
60 ! and Write Pointers set to zero, the Initial Mode
70 ! is set to FIFO Input with the Initial Lockout OFF.
80 !
90 Init(Fifo_in)
100 !
110 ! The Reference Word is programmed for the required
120 ! number of data words from the external device.
130 !
140 Write_ref_count(Fifo_in,1024)
150 !
160 ! The Memory card can now accept data from the external
170 ! device. Program execution is paused until the
180 ! external device has finished or a Timeout has occurred.
190 !
200 Wait_for(Fifo_in) ! wait for the "almost full"
210 ! condition to occur.
220 !
230 ! The data which has been stored in the Memory card
240 ! by the external device can now be read back into
250 ! the computer using the Input_rblock statement.
260 !
270 Input_rblock(Fifo_in,R_buffer(*))
280 !
290 END
```

69791A/69792A Memory Card System

The 69791A High Speed Memory card consists of two printed circuit boards. The two-board assembly can be installed in any two adjacent I/O slots of the 6944A Multiprogrammer mainframe. The first board (Memory card #1) is located in the lower numbered slot and is electrically connected to the second board (Memory card #2) by two flat cable assemblies. The 69791A is capable of storing up to 65,536 16-bit words at high speed. The 16-bit data can be written to, or read from, the card by either the computer or an external device. The computer transfers data via the Memory card's connection to the Multiprogrammer. External devices or other I/O cards transfer data via the external edge connector of Memory card #1. Since there is a different data path used by the computer than by external devices to communicate with the card, it is possible to simultaneously write data to and read data from the Memory card. This simple two-board Memory Card System can be expanded by the addition of one or more 69792A Memory Expansion cards.

The 69792A Memory Expansion card is a single printed circuit board that occupies one mainframe I/O slot. It is electrically connected to the 69791A via a special chaining cable (provided with each 69791A). Up to five 69792A Memory Expansion cards can be connected to a single 69791A by this chaining cable. Each Memory Expansion card that is added to the Memory Card System increases the number of 16-bit words that can be stored by 196,608. A storage capacity of up to 1,046,576 words can therefore be achieved with one 69791A High Speed Memory card and five 69792A Memory Expansion cards.

The Memory Card System has six modes of operation: FIFO (First In First Out) output, FIFO input, Recirculate output, Recirculate input, Simultaneous FIFO input, and Simultaneous Recirculate input. An output mode is used either to send computer data to the Memory Card System, or to transfer data from the Memory Card System to an external device. An input mode is used either to acquire data from an external device, or to read data into the computer from the Memory Card System. The Simultaneous mode is not used to transfer data between the computer and memory but, rather, to transfer data simultaneously to and from an external application. Data is read from an external device at the Input Data Lines and sent to an external device connected to the Output Data Lines. The difference between a FIFO mode and a Recirculate mode is simply whether you are dealing with a finite amount of data, or you want to continuously read or write data. The FIFO modes only allow data to be stored until memory is full. After that, no more can be stored until some data is read. Likewise, FIFO modes only allow reading stored words until memory is empty. After that, no more can be read until more data is stored. The Recirculate modes not only allow data to be stored until memory is full, but will continue to store data by overwriting the "oldest" memory locations. For example, once the last memory location is filled, the first memory location is overwritten. This wrap-around also occurs when reading data in Recirculate modes, allowing blocks of data to be read over and over again. A complete description of the Memory system and its modes of operation is given in the Model 69791A/69792A Operating Manual (HP Part No. 5960-1980).

DMA Transfers

The Memory Card System utilizes the Multiprogrammer backplane signal line enhancements which allow Direct Memory Access (DMA). This capability provides high speed data transfers from the Memory Card System directly into computer memory. This is initiated via an Input_rblock or Input_iblock statement if 98620B DMA Controller card is installed in the computer. If no DMA Controller card is present, data is transferred at a rate of 30 kHz. The DMA transfer rates for the Memory Card System can be summarized by the following chart:

	69791A Only	**69791A/69792A**
DMA Rate with no card edge input activity	220 kHz	200 kHz
DMA rate with card edge input activity	200 kHz	180 kHz

Menu Entires for the 69791A

Before programming the 69791A Memory Card System, you must specify various parameters in the Configuration Menu for the card. Following are descriptions of all the parameters required to configure this function. Many of the menu parameters, as well as show to use the menu keys, are described in detail in Chapter 3.

```
Name --- Memory                    Date: 21 Aug 1984
Model 69791A Memory System         Time: 17:40:39
File Name -- Chapter_9
User ID ----
Time-stamp - First Time

                                 Page 1 of 3

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot ..........

Number of Memory Extenders ... 0


Timeout ....... 5
```

Menu Page 1 of 3

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ..... 8
HP-IB Address ......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the Memory Card System. It has a legal range from 0 to 7. The default value is 0.

Slot

Identifies the location of the Memory card (circuit board #1) within the frame. Circuit board #2 is installed in the next highest slot,It has a legal range from 0 to 14. This parameter **MUST** be filled in; it has no default value.

Number of Memory Extenders

Enter the number of 69792A Memory Expansion cards that you have connected to the 69791A Memory card.

Timeout

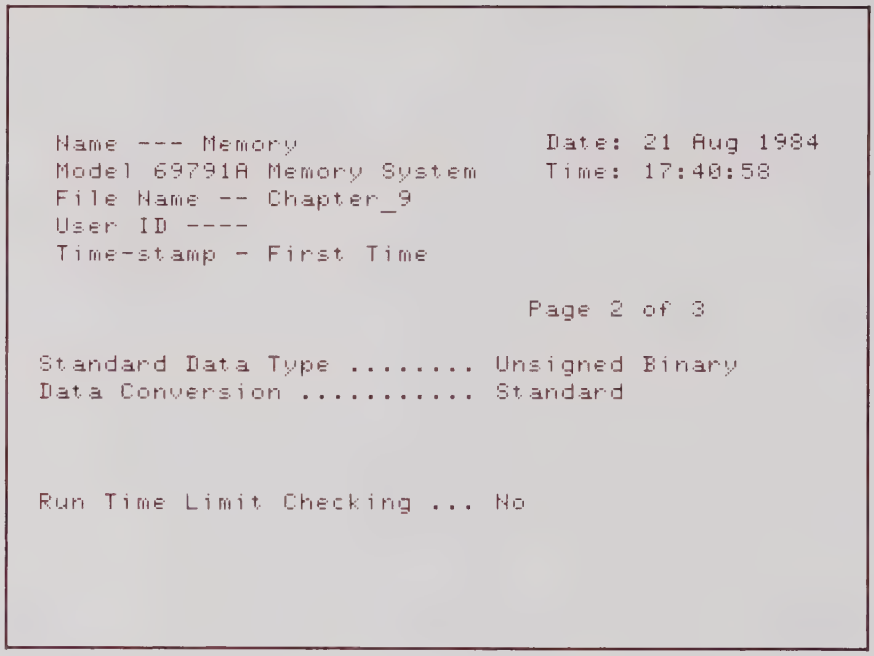
Specifies the time period in seconds that will be allowed for various operations, as listed below:

1. Output_rblock or Output_iblock statements — The time allowed for the specified number of words of the data in the array to be transferred from the computer into the Memory Card System. Program execution is suspended until the transfer is completed. If the data is not transferred within the time period specified, an error message (#803) is generated.
2. Input_rblock or Input_iblock statements — The time allowed for the specified number of words of data to be read into the computer array from the Memory Card System. Program execution is suspended until the transfer is completed. If the data is not transferred within the time period specified, an error message (#803) is generated.
3. Wait_for statement (FIFO input mode only) — The time allowed for the memory card to "fill up"; that is, for the number of words written into the card by an external device, but not yet read by the computer, to exceed the value of Reference Register 1. If a Wait_for statement does not detect this condition within the time period specified, error code 803 will be generated.
4. Wait_for statement (FIFO output mode only) — The time allowed for the memory card to "empty out"; that is, for the number of words remaining to be read from the card by an external device to become equal to, or less than, the value of Reference Register 1. If a Wait_for statement does not detect this condition within the time period specified, error code 803 will be generated.

The timeout period entered can be any number from 0.001 and 128,000. The default is 5 seconds.

NOTE

For Block transfers using DMA (direct memory access), the timeout value depends on the speed of the processor. Because Series 300 computers operate at clock frequencies greater than 8 MHz, for these computers the actual timeout will be less than that specified by the programmed value. Increase the timeout value to compensate for the computer's faster speed.


Name --- Memory
Date: 21 Aug 1984
Model 69791A Memory System
Time: 17:40:58
File Name -- Chapter_9
User ID ----
Time-stamp - First Time
Page 2 of 3
Standard Data Type ..... Unsigned Binary
Data Conversion ..... Standard
Run Time Limit Checking ... No

Menu Page 2 of 3

Standard Data Type — Specifies either "Unsigned Binary" or "2's Complement Binary" as the Data Type. The default selection is Unsigned Binary.

Unsigned Binary — The user reads or writes data in positive number only. The 14752A Output_rblock, and Input_rblock statements interpret these numbers to be unsigned binary values for the Memory card. The range of valid numbers if from 0 to 65535.

2's Complement Binary — The user can read or write data in positive or negative numbers. The 14752A Input_rblock and Output_rblock statements interpret the negative numbers as 2's complement binary value for the Memory Card System. The range of numbers that can be read or programmed is from −32768 to +32676. When using the Memory Card System with word lengths less than 16 bits, refer to the discussion in the previous 69790B Memory Card Section.

Data Conversion — Specifies how the data is to be scaled for the Input_rblock and the Output_rblock statements. One of two data conversion options can be selected: Standard and User. Note that for either an Output_iblock or an Input_iblock statement, the Data Conversion parameter does not apply; refer to the statement descriptions under "Advanced Programming" for information about the Output_iblock and Input_iblock statements.

"Standard" conversion for this card means that the data values in an Output__rblock or Input__rblock statement are in the form of unsigned binary or 2's complement values, whichever was selected as the standard data type. Standard (unsigned binary) is the default data conversion for this function.

"User" allows data to be specified in terms of the application rather than in terms of the card's Standard Units. As shown below, four additional entries appear on the menu when "User" conversion is selected.

User's P1 and P2

Standard P1 and P2

These values appear on the menu only when the User Convsersion option is selected. These parameters allow you to fill in the ratio of User's to Standard values so the 14752A can scale the data as it is sent to, or read from, the Memory Card System. The default values for P1 are 0; for P2 they are 1.

More About Data Conversion

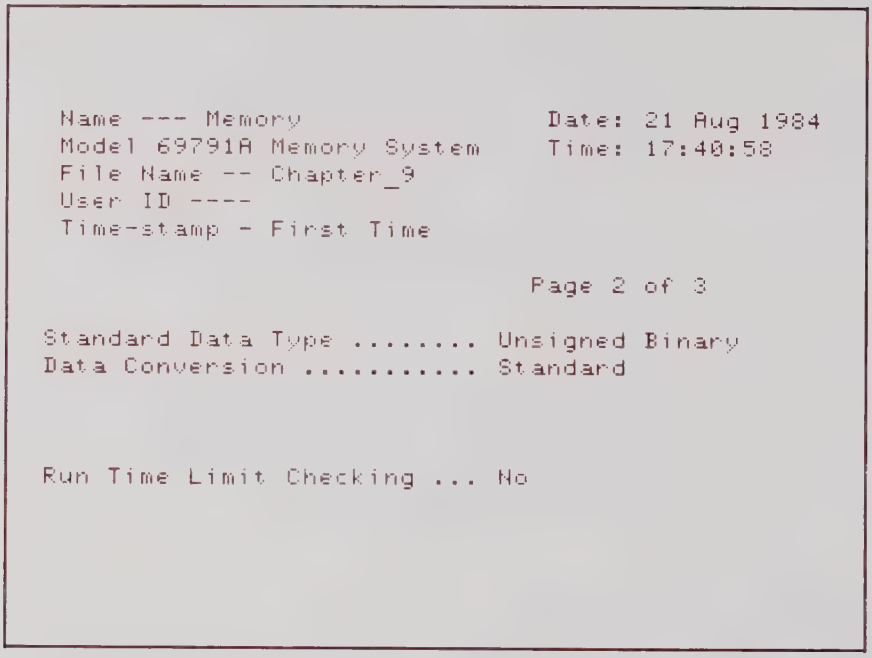
To illustrate the use of data conversion, assume we are using the Memory Card System to supply data to a Digital-to-Analog converter (DAC). The output of this converter is programmable from 0 to +20.480 volts for a digital input value from 0 to 65535. If "Standard" units were used, the data sent to the Memory Card System would be in the form of REAL numbers from 0 to 65535. We would like, however, to send data to the Memory Card System in terms of the output voltage of the DAC. Therefore, "User" units are selected in the menu. This will allow the program data to be in the form of output voltage and the 14752A will convert this data into the data format required by the DAC.

When "User" conversion is selected in the menu, four other parameters must also be specified. These parameters give the 14752A all the data it needs to scale the "Standard" units to "User's" units (Volts). To perform the conversion, it is necessary to specify two points, for instance the minimum output and the maximum output. These two points will specify the two parameters for the 14752A. In this example, the minimum output of 0 volts equals a data value of 0 and the maximum output of 20.480 volts equals a data value of 65535. The range of values would be as follows:

	minimum output (P1)	**maximum output (P2)**
Standard Units	0	65535
User's Units	0 volts	20.480 volts

From this list we can see that: Standard P1 = 0
Standard P2 = 65535
User's P1 = 0
User's P2 = 20.480

While User's Units are more convenient to use, it takes the 14752A 20% to 30% longer to fill the card with this data than if Standard units were specified.


Name --- Memory
Date: 21 Aug 1984
Model 69791A Memory System
Time: 17:40:58
File Name -- Chapter_9
User ID ----
Time-stamp - First Time
Page 2 of 3
Standard Data Type ..... Unsigned Binary
Data Conversion ..... Standard
Run Time Limit Checking ... No

Menu Page 2 of 3 (continued)

Menu Page 2 of 3 (continued)

Run Time Limit Checking

If YES is selected, values sent to the Memory Card System using the Output_rblock statement will be checked against the specified High and Low Limits. Values which are read from the Memory card by the computer, or values that are sent to it by an "Output_iblock" statement, are not checked against either the High Limit or the Low Limit. This optional parameter can be toggled on or off, by selecting YES or NO. The default is NO limit checking. When Run Time Limit checking is selected, the following two parameters **MUST** be filled in; they have no default.

High Limit

Is used only when the Run Time Limit Checking is ON. Specifies the upper limit for the data values specified in the Output_rblock statement. The value entered must be greater than the Low Limit specified. An error (#802) will be generated when a programmed output value is greater than this limit. This value **MUST** be entered in the same units that are selected for the programming units.

Low Limit

Is used when the Run Time Limit Checking is ON. Specifies the lower limit for the data values specified in the Output_rblock statement. The value entered must be less than the High Limit specified. An error (#802) will be generated when a programmed output value is less than this limit. The value **MUST** be entered in the same units that are selected for the programming units.

```
Name --- Memory                    Date: 21 Aug 1984
Model 69791A Memory System         Time: 17:41:09
File Name -- Chapter_9
User ID ----
Time-stamp - First Time

                                 Page 3 of 3


Initial Read Pointer ..... 0
Initial Write Pointer ..... 0
Initial Reference Register 1 ..... 0
Initial Reference Register 2 ..... 0
Initial Mode ..... FIFO In
Initial Lockout ..... On

Level Interrupt ..... Disabled
Error Interrupt ..... Disabled
```

Menu Page 3 of 3

Initial Read Pointer

Specifies the memory location from which the first data value will be read. The legal range for this parameter is from 0 to the last memory location with a default value of 0.

Initial Write Pointer

Initial Write Pointer Specifies the memory location where the first data value will be written. The legal range for this parameter is from 0 to the last memory location. The default value is 0.

Initial Reference Register 1

Specifies the value programmed into the Reference Register 1 on the Memory card when an Init or System_init statement is programmed. The function of this entry depends upon the mode of operation that is in effect:

1. In the FIFO Input mode Reference Register 1 sets the number of words that will be stored into the Memory Card System by the external device before it will indicate that memory is "almost full". *

2. In the FIFO Output mode Reference Register 1 sets the number of words that remain in the card before it will indicate that it is "almost empty". *

3. In the Recirculate modes, Reference Register 1 truncates the size of memory that can be read by setting a limit for the last addressable memory location. Note, Reference Register 1 only effects the reading of data NOT the storing of data.

* Indication of "almost empty" or "almost full" can be checked with the statements, Check_done, Wait_for, Get_conditions, and Enable_intr. All of these statements are described in the Advanced Programming section later in this chapter. The "almost full" and "almost empty" conditions are also indicated by the setting of a signal called EOP that is located at the edge connector (J2-n) of Memory card No. 1.

Initial Reference Register 2

The value entered here will be sent to Reference Register 2 when an Init or System_init statement is executed. Reference Register 2 sets a limit on the lowest memory location that can be read. It should be programmed to zero while operating in any of the FIFO modes because it only limits the readable locations and not the storeable ones. It is most useful in the Recirculate Output mode when sending data to an external device.

Initial Mode

Either the FIFO Output, FIFO Input, FIFO Simultaneous Input, Recirculate Output, Recirculate Input or Recirculate Simultaneous Input mode can be selected. The default mode is FIFO Input.

Initial Lockout

The Memory Card System can be inhibited from either inputing or outputing data from/to an external device by turning the External Lockout ON. Lockout is used when you want to transfer data in or out of the computer without the interruption of an external device. The lockout can be either turned ON or OFF. The default is Lockout ON.

The following two menu entries establish initial interrupt mask values. The default for both of these entries is "Disabled". If interrupts are not going to be used in your application, you do not have to make any entries for these parameters. If you will be using interrupts, you must enable at least one of the two types. To get any interrupts from the Memory Card System you must use the Enable_intr statement that is described later under Advanced Programming.

Level Interrupt

When enabled, allows interrupts to be generated for either of the following two conditions:

a. When the memory becomes "almost empty" in the FIFO Output mode. That is, when the number of words remaining in memory becomes equal to, or less than, the value that was programmed to Reference Register 1.

b. When memory becomes "almost full" in the FIFO Input mode. That is, when the number of words written into memory by an external device, but not yet read by the computer, becomes greater than the value that was programmed to Reference Register 1.

Error Interrupt

When enabled, allows interrupts to be generated for either of the following two error conditions:

a. When an external device attempts to write data to a full memory. This is called a "Missed Data" type interrupt.

b. When an attempt is made to read data from an empty memory. This is called an "Invalid Read" type interrupt.

Programming Fundamentals

Only a few fundamental statements are needed to program the Memory Card System. The following is a summary of these statements. For details concerning the syntax requirements for the Name and Value parameters, refer to Chapter 3.

Statement	Description
Init(Name)	Loads the card with the initial values for mode, external lockout, Reference Registers, Read Pointer, Write Pointer, Difference Count, and Timeout as specified in the Configuration Menu. This instruction also disables interrupts from the Memory Card System if they have been enabled with the advanced programming instruction, Enable_intr. Also clears latched status bits 4 and 5.
Input_rblock(Name, Array(*)[,Count [,Index]])	Reads a block of data from the Memory Card System, converts the data according to the Standard Data type and Data Conversion specified, and returns the data in the REAL array called "Array". The card must be in either the FIFO input or Recirculate input mode; if it is not, an error message (code 825) will be generated. The number of data words read from the Memory Card System is equal to the size of the data array. That is, if the data array is dimensioned for 1000 then 1000 data words will be returned. Optionally, if the number

of data words to be read is less than the size of the data array then a parameter, referred to as "Count", can be specified for the required number of data words. If "Count" is used, you can also use the optional parameter, "Index". Index can be used with the Count parameter to specify the position in the array where the first input data word will be placed; subsequent data words are placed in the successive array positions that follow. The data array **MUST** be previously DIMensioned or ALLOCATEd in the user's program, otherwise a BASIC system error (#13) will be generated.

NOTE

If your computer has cache memory, you may get a BASIC error (388) when using this statement with DMA transfers. In this case, turn off cache memory before executing the statement and turn it on again after the statement. The command to turn cache memory off is:

CONTROL 32,1;0

The command to turn cache memory on is:

CONTROL 32,1;1

Output_rblock(Name, Array(*)[,Count [,Index]])

Converts a block of REAL data according to the Standard Data Type and Data Conversion that was selected in the menus and then sends the block to the Memory card. The card must be in either the FIFO output or Recirculate output mode; if it is not, an error message (code 825) will be generated. The number of words written to the Memory card is equal to the size of the data array called "Array". That is, if the data array is dimensioned for 2048 words, then 2048 words will be written to the Memory Card System.

Optionally, if the number of data words to be sent is less than the size of the data array then the parameter, "Count", can be specified for the required number of data words that will be sent to the Memory card. The optional parameter, Index can be used with the Count Parameter to specify the array position of the first data word that will be sent. If Index is not specified, it defaults to 1. The data array **MUST** be previously DIMensioned or ALLOCATEd in the user's program, otherwise a BASIC system error (#13) will be generated.

The operating mode of the Memory Card system can be changed from the one selected in the Configuration phase by using one of the following six statements.

Statement	Description
Set_fifo_in(Name)	Changes the Memory Card System's operating mode to the FIFO Input mode.
Set_fifo_out(Name)	Changes the Memory card's operating mode to the FIFO Output mode.
Set_recirc_in(Name)	Sets the Memory Card System's operating mode to the Recirculate Input mode.
Set_recirc_out(Name)	Sets the Memory Card System's operating mode to the Recirculate Output mode.
Set_fifo_simul(Name)	Sets the Memory Card System's operating mode to the Simultaneous FIFO Input mode.
Set_rec_simul(Name)	Sets the Memory Card System's operating mode to the Simultaneous Recirculate Input Mode.

Example #1 FIFO Input

One of the ways in which the 69791A Memory Card System can be used is a simple data buffer. Suppose that we wish to store a number of data words generated by an external device and read them back into the computer. No data conversion will be required in this example; only 16-bit words will be transferred from the external device through the Memory Card System to the computer.

Programming Example for a Simple Data Input

```
 10 COM/Names/Fifo_in
 20 Configure("Ask_me","MYFILE")
 30 REAL R_buffer(6000)
 40 !
 50 ! The Memory Card System is initialized with the Read
 60 ! and Write Pointers set to zero, Initial Mode
 70 ! set to FIFO Input and the Initial Lockout OFF.
 80 !
 90 Init(Fifo_in)
100 !
110 ! The Memory Card System can now accept data from the
120 ! external device. Assume that the device
130 ! requires 3 seconds to generate 6000 data words.
140 !
150 WAIT 3 ! wait for data
160 !
170 ! The data has been stored in the Memory Card System
180 ! by the external device and can now be read back into
190 ! the computer using the Input_rblock statement
200 !
210 Input_rblock(Fifo_in,R_buffer(*))
220 !
230 END
```

Example #2 FIFO Output

The 69791A Memory Card System is bi-directional, that is, it is not only an input device but also an output device. The previous example showed how to read data from the Memory Card System into the computer. This example will illustrate how to output a block of REAL data to the Memory Card System. In this example assume that a data array has been dimensioned and that this array is used by different routines in the program. In other words, it is a general purpose array. Only the first 5000 data words in this array are to be output to the Memory Card System. This requires the use of an optional parameter, called "Readings", in the Output_rblock statement.

Programming Example for a Simple Data Output

```
 10 COM/Names/Fifo_out
 20 Configure("Ask_me","MYFILE")
 30 !
 40 ! General purpose array, R_buffer, is
 50 ! dimensioned and the optional parameter,
 60 ! Readings is specified as a REAL variable
 70 !
 80 REAL R_buffer(6000)
 90 REAL Readings
100 !
110 ! The Memory Card System is initialized with the Mode
120 ! set to FIFO Output, Initial Lockout OFF, and the
130 ! Read and Write Pointers set to zero. Since the data
140 ! will be in User Units or REAL numbers, data conversion
150 !  must be specified.
160 !
170 Init(Fifo_out)
180 !
190 ! Assume that the data to be output to the Memory
200 ! card had been stored in the first 5000 positions of the
210 ! array, R_buffer. The variable, Readings, is set to the
220 ! number of words which are to be output to the card.
230 !
240 Readings=5000
250 !
260 ! The first 5000 data words contained in the array are
270 ! now output to the Memory Card System.
280 !
290 Output_rblock(Fifo_out,R_buffer(*),Readings)
300 !
310 ! Now lets use Index to send the next 100 values of the
320 ! array, starting with array element 5001:
330 !
340 Output_rblock(Fifo_out,R_buffer(*),100,5001)
350 END
```

Example #3 Recirculate Output

This example shows how Reference Register 1 and Reference Register 2 are used to set the lowest and highest memory location that is to be read. Suppose that we want to generate a sinusoidal waveform by sending data to a Model 69720A D/A card. The data for one complete cycle is contained in memory locations 200 through 4000. Each data word is paced out of memory by a time base that is connected to the $\overline{\text{DACOUT}}$ input at the edge connector of Memory Card 1. For this example must configure the Timebase to generate the desired frequency as described in the Timer section of this chapter. Also, we will use all of the default menu entries for the Memory Card System except for:

Initial Reference Register 1 = 200
Initial Reference Register 2 = 4000
Initial Lockout = OFF

Programming Example Using Recirculate Output Mode

```
 10 COM/Names/Memory,Timebase
 20 Configure("Ask_me","MYFILE")
 30 System_init
 40 !
 50 ! Assume that the entire 65536 memory locations contain
 60 ! several cycles of a digitized sinusoidal wave. In
 70 ! order to re-create the waveform we must repeatedly send
 80 ! one complete cycle to a Model 69720A D/A card. Each data
 90 ! word is read from memory at a fixed rate. The frequency of
100 ! the generated waveform is determined by the period rate
110 ! of the Timebase. First set the Recirculate Output mode.
120 !
130 Set_recirc_out(Memory)
140 !
150 ! Now we can start sending the data to the
160 ! D/A card by turning on the time base.
170 !
180 Start(Timebase)
190 !
200 END
```

Advanced Programming

There are 14752A statements which allow special purpose programming of the Memory function. They are considered advanced programming statements because you need a good understanding of the Multiprogrammer I/O cards and system to put together a meaningful program. In general, these statements are used only when precise control of the card's data flow is a concern.

They allow features such as: interrupts, reading from and writing to the card registers, and checking card status. Most of these features are explained in greater detail in Chapter 10. The following is a list of the advanced programming statements which can be used for the 69791A Memory function.

The following two statements, Input_iblock and Output_iblock, do not format the data according to the data conversion option selected; that is, the data is returned in exactly the form that it was stored on the Memory card. These two statements can be used to speed up the data transfer rate between the Memory card and the computer.

Statement	Description
Input_iblock(Name, Iarray(*)[,Count [,Index]])	Similar to the Input_rblock statement. Reads a block of 16-bit 2's complement data from the Memory Card System and returns this data in the INTEGER array called "Iarray". The card must be in either the FIFO input or the Recirculate input mode; if not, an error message (#825) will be generated. The number of data words read from the Memory card is equal to the size of the data array, that is, if the data array is dimensioned for 1000 then 1000 data words will be returned. The optional parameter, Count, specifies the number of data words that will be read from the Memory card. The value of Count must be less than, or equal to the size of the data array. The optional parameter, Index, can be used with the Count parameter to specify the array location where the first data word will be placed. The data array **MUST** be previously DIMensioned or ALLOCATEd in the user's program otherwise a BASIC system error (#13) will be generated.

NOTE

If your computer has cache memory, you may get a BASIC error (388) when using this statement with DMA transfers. In this case, turn off cache memory before executing the statement and turn it on again after the statement. The command to turn cache memory off is:

CONTROL 32,1;0

The command to turn cache memory on is:

CONTROL 32,1;1

Output_iblock(Name, Iarray(*)[,Count [,Index]])

This statement is similar to the Output_rblock statement. It writes a block of 16-bit 2's complement INTEGER data to the Memory card. The card must be in either the FIFO outout mode or the Recirculate output mode; if it is not, an error message (#825) will be generated. The optional parameter, Count, can be used to specify the number of data words to be sent to the card. If Count is not specified, the number of words written to the Memory card is equal to the size of the data array. That is, if the data array is dimensioned for 2000 words then the first 2000 words of the array will be written to the card. The optional parameter, Index, can be used with Count to specify the array location of the first data word that will be sent. The data array **MUST** be previously DIMensioned or ALLOCATEd in the user's program, otherwise a BASIC system error (#13) will be generated.

Translate(Name, Iarray(*),User(*))

Used to process data acording to the data conversion specified on page 2 of the 69791A/69792A configuration menu. This statement enables you to seperate the data conversion process from the actual data transfer. If the memory is in an input mode, the INTEGER data in "Iarray" is converted to REAL numbers and placed into the "User" array. If the memory is in an output mode, the REAL data in the User array is converted to INTEGER data and placed into Iarray. Arrays User and Iarray **MUST** be previously DIMensioned or ALLOCATEd in the user's program, otherwise a BASIC system error (#13) will be generated.

Check_done(Name,Flag)

Inquires about the status of the Memory card's operation. Flag returns a value of one if:

1. In the FIFO Input mode, the number of words written to the Memory card by an external device, but not read by the computer, is greater than the value stored in Reference Register 1. This is the "almost full" condition.

2. In the FIFO Output mode, the number of words to be read from the Memory Card System, by an external device, is equal to, or less than, the value stored in Reference Register 1. This is the "almost empty" condition.

If neither of these conditions occur, the variable, Flag, is set to zero. Also, if the Memory card is in either the Recirculate Input or the Recirculate Output mode and a Check_done statement is used, an error (#803) is generated.

Wait_for(Name)

Like the Check_done statement, Wait_for also checks the cards status in the FIFO modes; however, program execution is suspended until one of the two conditions, described above, occur. If neither of the two conditions occur before the Timeout value has elapsed, an error message (#803) is generated. This statement is used if you want to wait for the Memory Card System to indicate an "almost full" or "almost empty" condition while in either the FIFO input or FIFO

Advanced Programming

There are 14752A statements which allow special purpose programming of the Memory function. They are considered advanced programming statements because you need a good understanding of the Multiprogrammer I/O cards and system to put together a meaningful program. In general, these statements are used only when precise control of the card's data flow is a concern.

They allow features such as: interrupts, reading from and writing to the card registers, and checking card status. Most of these features are explained in greater detail in Chapter 10. The following is a list of the advanced programming statements which can be used for the 69791A Memory function.

The following two statements, Input_iblock and Output_iblock, do not format the data according to the data conversion option selected; that is, the data is returned in exactly the form that it was stored on the Memory card. These two statements can be used to speed up the data transfer rate between the Memory card and the computer.

Statement	**Description**
Input_iblock(Name, Iarray(•)[,Count [,Index]])	Similar to the Input_rblock statement. Reads a block of 16-bit 2's complement data from the Memory Card System and returns this data in the INTEGER array called "Iarray". The card must be in either the FIFO input or the Recirculate input mode; if not, an error message (#825) will be generated. The number of data words read from the Memory card is equal to the size of the data array, that is, if the data array is dimensioned for 1000 then 1000 data words will be returned. The optional parameter, Count, specifies the number of data words that will be read from the Memory card. The value of Count must be less than, or equal to the size of the data array. The optional parameter, Index, can be used with the Count parameter to specify the array location where the first data word will be placed. The data array **MUST** be previously DIMensioned or ALLOCATEd in the user's program otherwise a BASIC system error (#13) will be generated.
Output_iblock(Name, Iarray(•)[,Count [,Index]])	This statement is similar to the Output_rblock statement. It writes a block of 16-bit 2's complement INTEGER data to the Memory card. The card must be in either the FIFO outout mode or the Recirculate output mode; if it is not, an error message (#825) will be generated. The optional parameter, Count, can be used to specify the number of data words to be sent to the card. If Count is not specified, the number of words written to the Memory card is equal to the size of the data array. That is, if the data array is dimensioned for 2000 words then the first 2000 words of the array will be written to the card. The optional parameter, Index, can be used with Count to specify the array location of the first data word that will be sent. The data array **MUST** be previously DIMensioned or ALLOCATEd in the user's program, otherwise a BASIC system error (#13) will be generated.

Advanced Programming (continued)

Statement	Description
Translate(Name, Iarray(*),User(*))	Used to process data acording to the data conversion specified on page 2 of the 69791A/69792A configuration menu. This statement enables you to seperate the data conversion process from the actual data transfer. If the memory is in an input mode, the INTEGER data in "Iarray" is converted to REAL numbers and placed into the "User" array. If the memory is in an output mode, the REAL data in the User array is converted to INTEGER data and placed into Iarray. Arrays User and Iarray **MUST** be previously DIMensioned or ALLOCATEd in the user's program, otherwise a BASIC system error (#13) will be generated.
Check_done(Name,Flag)	Inquires about the status of the Memory card's operation. Flag returns a value of one if: 1. In the FIFO Input mode, the number of words written to the Memory card by an external device, but not read by the computer, is greater than the value stored in Reference Register 1. This is the "almost full" condition. 2. In the FIFO Output mode, the number of words to be read from the Memory Card System, by an external device, is equal to, or less than, the value stored in Reference Register 1. This is the "almost empty" condition. If neither of these conditions occur, the variable, Flag, is set to zero. Also, if the Memory card is in either the Recirculate Input or the Recirculate Output mode and a Check_done statement is used, an error (#803) is generated.
Wait_for(Name)	Like the Check_done statement, Wait_for also checks the cards status in the FIFO modes; however, program execution is suspended until one of the two conditions, described above, occur. If neither of the two conditions occur before the Timeout value has elapsed, an error message (#803) is generated. This statement is used if you want to wait for the Memory Card System to indicate an "almost full" or "almost empty" condition while in either the FIFO input or FIFO output mode, respectively. The Wait_for statement is only useful in either of these two FIFO modes; an error message (#803) will be generated if the Memory Card System is in one of the Recirculate modes and a Wait_for statement referencing the Memory Card System is encountered in the program.
Read_rpointer(Name, Value)	Returns the address of the next memory location to be read from. The address is placed in the variable, Value.

Advanced Programming (continued)

Statement	Description
	output mode, respectively. The Wait_for statement is only useful in either of these two FIFO modes; an error message (#803) will be generated if the Memory Card System is in one of the Recirculate modes and a Wait_for statement referencing the Memory Card System is encountered in the program.
Read_rpointer(Name, Value)	Returns the address of the next memory location to be read from. The address is placed in the variable, Value.
Read_wpointer (Name,Value)	Returns the address of the next memory location to be written into. The address is placed in variable, Value.
Read_ref_reg1	Returns the value stored in Reference Register 1 of the Memory card. The value is placed in variable, Value.
Read_ref_reg2 (Name,Value)	Returns the value stored in Reference Register 2 of the Memory card. The value is placed in variable, Value.
Read_dif_count (Name,Value)	Returns the difference between the Memory Card System's Read and Write Pointers. The difference is placed in variable, Value.
Write_rpointer(Name, Value)	Sets the Memory Card System's Read Pointer to the address in the variable, Value, and adjusts the value of the Difference Count. The range of legal addresses is 0 through the last memory location.
Write_wpointer (Name,Value)	Sets the Memory Card System's Write Pointer to the address in the variable, Value and adjusts the value of the Difference Count. The range of legal addresses is 0 through the last memory location.
Write_ref_reg1 (Name,Value)	Sets Reference Register 1 equal to the number contained in the variable, Value. The range of legal values is 0 through to the last memory location.
Write_ref_reg2 (Name,Value)	Sets Reference Register 2 equal to the number contained in the variable, Value. The range of legal values is 0 through to the last memory location.
Write_dif_count (Name,Value)	Sets the Memory Card System's Difference Count register equal to the number in the variable, Value. Note, the Difference Count is adjusted (to reflect the true difference between the write and read pointers) every time that either the Read or the Write pointer is changed. So, the value you write to the Difference Count will be changed if either the Read or Write pointer is written to. The range of values is 0 through the last memory location.

Advanced Programming (continued)

Statement	Description
Enable__lockout(Name)	Prevents data from being written to, or read from, the Memory Card System by an external device. Used to prevent an external device from changing any data or pointers.
Disable__lockout(Name)	Allows the Memory card to respond to write/read requests from an external device. The Disable__lockout statement does this by enabling the Memory Card System's edge connector handshake lines.
Get__conditions(Name, Value)	Returns the status of the Memory Card System and clears latched status bits 6 and 7. The variable Value is assigned a binary weighted value as follows:

Value	Condition
1	EOP is set.
2	Card is BUSY (Full in input mode; empty in output mode).
4	Error interrupts are masked.
8	Level interrupts are masked.
16	Attempt was made to write data to a full memory.
32	Attempt was made to read data from an empty memory.
64	A RAM read cycle is in progress.
128	A RAM write cycle is in progress.

Set__level__ien(Name,Flag) If Flag equals 1, the level interrupts will be enabled by the Enable__intr statement. If Flag equals 0, level interrupts are disabled by the Enable__intr statement.

Set__error__ien(Name,Flag) If Flag equals 1, the error interrupts will be enabled by the Enable__intr statement. If Flag equals 0, error interrupts are disabled by the Enable__intr statement.

Enable__intr(Name,Vector) Allows the 69791A Memory Card System to signal the computer via the interrupt system when either the "almost empty" or "almost full" condition occurs. The Memory card can be programmed to interrupt the computer only when it is in either the FIFO input mode or the FIFO output mode. The conditions under which the Memory Card System will interrupt the computer are:

1. In the FIFO Input mode, the computer will be signalled when the number of words written to the Memory Card System by the external device but not read from it by the computer is greater than the value stored in Reference Register 1 (the "almost full"

Advanced Programming (continued)

Statement **Description**

condition). For example, suppose that when 2500 words are stored in the Memory Card System by an external device, the computer should be informed to read them. Reference Register 1 is set equal to the number of data words required or 2500. The computer is informed when data word #2500 is read into the Memory Card System. The computer can then read the entire 2500 word data block.

2. In the FIFO Output mode, the computer will be signalled when the number of words yet to be read from memory by an external device is equal to or less than the value stored in Reference Register 1 (the "almost empty" condition). For example, suppose several large blocks of data are to be transferred from the computer to an external device. Reference Register 1 is set to a value that will allow the computer to send a block of data to the Memory card before the external device can read the last data word from the Memory card. The computer is signalled when the number of data words to be read from the Memory Card System becomes equal to the value in Reference Register 1. The computer will then send a new block of data to the Memory Card System. The process will continue until all the blocks of data in the computer have been sent to the external device.

When the function interrups the computer, a variable called "Reason" is passed to the interrupt routine. The variable, Reason, has a binary weighted value that indicates the following cause(s) for the interrupt:

Value	Condition
1	The data level has been reached(Level interrupt)
2	Data was sent to a full memory (Missed data interrupt)
4	An attempt was made to read data from an empty memory (Invalid Read Interrupt)

The Enable__intr statement must be used in conjunction with the interrupt techniques as described in Chapter 10.

Disable__intr(Name) Inhibits the Memory Card System from signalling the computer via the interrupt system. Note that this statement disables both Level and Error interrupt sources but does not affect the interrupt mask. Meaningful only when the Enable__intr statement has already been programmed. Also note that the Init statement will also disable interrupts.

Set__timeout(Name,Value) Changes the value of timeout from the time specified in the menu to the new Value. The range of legal values is 0.001 to 128,000 seconds.

Advanced Programming Example

To illustrate some of the advanced programming features, Example #1 of the Fundamental Programming section, has been modified for the example below. The Write_ref_reg1 and the Wait_for statements are used instead of the BASIC WAIT statement. In this way, program execution will be paused only for as long as is needed for the external device to write the required amount of data onto the Memory card. Also, the program will wait any additional time that the the external device may require. A Timeout value should be set during the Configuration phase to allow enough time for the external device to load the data into the card.

Advanced Programming Example for Data Input

```
 10 COM/Names/Fifo_in
 20 Configure("Ask_me","MYFILE")
 30 REAL R_buffer(6000)
 40 !
 50 ! The Memory Card System is initialized with the Read
 60 ! and Write Pointers set to zero, the Initial Mode
 70 ! is set to FIFO Input with the Initial Lockout OFF.
 80 !
 90 Init(Fifo_in)
100 !
110 ! Reference Register 1 is programmed so that the required
120 ! number of data words from the external device exceeds
130 ! the Difference Count.
140 !
150 Write_ref_reg1(Fifo_in,5999)
160 !
170 ! The Memory Card System can now accept data from the external
180 ! device. Program execution is paused until the
190 ! external device has finished or a Timeout has occurred.
200 !
210 Wait_for(Fifo_in) ! wait for the "almost full"
220 ! condition to occur.
230 !
240 ! The data which has been stored in the Memory Card System
250 ! by the external device can now be read back into
260 ! the computer using the Input_rblock statement.
270 !
280 Input_rblock(Fifo_in,R_buffer)
290 !
300 END
```

Buffer Function

The Buffer menu has been provided for all multiple card functions involving the 69790B Memory card or 69791A/792A Memory Card System. A multiple card function is a combination of two or more I/O cards working together as a "subsystem". For example, the Memory card or Memory Card System can be combined with a front-end 69720A/69721A D/A card to form a Buffered D/A multiple card function, or a front-end 69751A/69759A A/D card to form a Buffered A/D multiple card function. For card interconnections refer to Appendix C.

In the Buffered D/A multiple card function, the Memory card or Memory Card System serves as a data buffer which frees the computer from the task of sending data directly to the D/A card. The computer instead preloads the buffer memory with all the required data values which will be output to the D/A card. Without this buffering or storing of data, the speed of the waveform generation would be slowed considerably.

In the Buffered A/D multiple card function, the buffer memory frees the computer from the task of reading data directly from the A/D card and allows the function to operate at a rate independent of the computer's speed. Thus, the Buffered A/D Converter function can sample,convert, and store voltage readings while the the computer is performing other tasks. The multiplexed A/D function is a special application of the Buffered A/D function where several 69759A A/D cards can store data into one 69791A Memory Card System (refer to 69759A Operating Manual HP Part No. 69759-90001 for hardware details).

Once the cards have been linked together with cable hardware, they need to be initialized (via the Init statement) as a subsystem in preparation for a measurement. The software provides a means to do this. In order to implement this feature, the user must specify how the cards are cabled together by supplying the name of the front-end card in the Buffer menu. This establishes a link between the Memory card or Memory Card System and the front-end input or output card. This means that a single Init statement using the Buffer name, such as Init(Buffer), will initialize all of the cards in the multiple card configuration.

The major purpose of the Buffer menu is to allow the Memory card or Memory Card System to retrieve the data conversion information from the front-end card's menu. For this reason, the data conversion option shown in the 69790B Memory card and 69791A Memory Card System menus has been eliminated from the Buffer menu. This is most useful for the multiplexed A/D scheme where a group of 69759A A/D cards supply information to one 69791A Memory Card System. In this case, the multiplexed data is post processed using the data conversion and range information specified for each particular A/D card. There are two program statements provided for such processing of data; Translate and Demultiplex. Translate has been provided for all applications involving the Buffer menu and Demultiplex only for multiplexed applications. For comprehensive examples involving the Buffer menu, refer to Application Note 316-5 on data capture.

Menu Entries For Buffer

Before programming the Buffer menu, you must specify various parameters for the Memory card or Memory Card System, in the Configuration menu.You must also specify the front-end card being used in the application. Following are descriptions of all the parameters required to configure this function. More information about the parameters,and about the Memory card or Memory Card System in general, can be found in the 69790B and 69791A/69792A sections. It is recommended that this be read before configuring the Buffer menu.

```
Name --- Buffer                    Date: 21 Aug 1984
Model Buffer                       Time: 17:33:13
File Name -- Chapter_9
User ID ----
Time-stamp - First Time

                                 Page 1 of 2

 Multiprogrammer Type ... HP6944A
 Interface Select Code .. 29

Frame ......... 0
Slot ..........

Initial Mode .......... FIFO
Initial Lockout ....... On

Timeout ....... 5
```

Menu page 1 of 2

Multiprogrammer Type

Identifies the model of the Multiprogrammer. The default is the 6944A. The Interface Select Code for the 6944A identifies the computer select code of the 98633A Multiprogrammer Interface Card. The legal range is from 8 to 31. The default is 29.

For Multiprogrammer type 6942A, the following information appears on the screen.

```
Multiprogrammer Type ... HP6942A
HP-IB Select Code ..... 8
HP-IB Address ......... 23
```

The HP-IB Select Code identifies the computer select code of the 98624 HP-IB Interface. The range is from 8 to 31. The default is 8. The HP-IB address identifies the address of the 6942A Multiprogrammer. The range is from 0 to 30. The default is 23.

Frame

Specifies the Multiprogrammer frame which contains the 69790B Memory card or 69791A Memory Card System. The legal is from 0 to 7. The default is 0.

Slot

Identifies the location to the 69790B Memory card or 69791A Memory Card System within the specified frame(card #1 of the two card set). For the 69791A Memory Card System, the slot(s) of the 69792A extender card(s) do not have to be specified. This parameter has no default. It **MUST** be filled in.

Multi Interface

Identifies the computer select code of the 98633A Multiprogrammer Interface card. The legal range is from 8 to 31. The default is 29.

Initial Mode

Either FIFO or Recirculate can be selected. FIFO mode specifies that memory will act like a first in-first out buffer. Recirculate specifies that memory will act like a circular buffer. The default is FIFO.

Initial Lockout

The Memory card or Memory Card System can be inhibited from either receiving/ sending data from/to an external device by turning the External Lockout ON.Lockout is used when you want to transfer data in or out of the computer without the interruption of an external device. The Lockout can either be turned ON or OFF. The default is ON.

Timeout

Specifies the time period in seconds that will be allowed for various operation, as listed below:

1. **Output_rblock or Output_iblock statements (FIFO output mode only).** The time allowed for the specified number of words of the data in the array to be transferred from the computer into the Memory card or Memory Card System. Program execution is suspended until the transfer is completed. If the data is not transferred within the time period specified, an error message (#803) is generated.
2. **Input_rblock or Iput_iblock statements (FIFO input mode only).** The time allowed for the specified number of words of data to be read into the computer array from the Memory card or Memory Card system. Program execution is suspended until the transfer is completed. If the data is not transferred within the time period specified, an error message (803) is generated.
3. **Wait_for statement (FIFO output mode only).** The time allowed for the Memory card or Memory Card System to "empty out", that is, for the number of words remaining to be read from the card by an external device to become equal to, or less than, the value of the Reference Register 1 (Ref Reg1). If a Wait_for statement does not detect this condition within the time period specified, error message (#803) is generated.
4. **Wait_for statement (FIFO input mode only).** The time allowed for the Memory card or Memory Card System to "fill up"; that is, for the number of words written into the card by an external device, but not yet read by the computer, to exceed the value of the Reference Count or Reference Register 1 (Ref Reg1). If a Wait_for statement does not detect this condition within the time period specified, error message (#802) is generated.

The timeout period entered can be any number from 0.001 and 128,000. The default is 5 seconds. Timeout does not apply to a 69790B Memory card or 69791A Memory Card System that is in either the recirculate input or recirculate output mode.

NOTE

For Block transfers using DMA (direct memory access), the timeout value depends on the speed of the processor. Because Series 300 computers operate at clock frequencies greater than 8 MHz, for these computers the actual timeout will be less than that specified by the programmed value. Increase the timeout value to compensate for the computer's faster speed.

```
Name --- Buffer                    Date: 21 Aug 1984
Model Buffer                       Time: 17:33:30
File Name -- Chapter_9
User ID ----
Time-stamp - First Time

                                  Page 2 of 2
Memory Type .... 69790B
Reference Count .... 4095
Buffer Direction .. In   Front End Type .. 69751A
A/D Name ....
```

Menu Page 2 of 2

Memory Type

Specifies whether a 69790B Memory card or 69791A Memory Card System is being used as the data buffer in the multiple card function. The default is 69790B. *

No.of Extenders

Appears only when the Memory Type is 69791A. This specifies how many 69792A Memory Expansion cards are being used with the 69791A. The maximum amount possible is 5 (increases memory capacity to 1M byte). The default is 0.

Ref Reg 2

Appears only when the Memory Type is 69791A. This specifies the value programmed into Reference Register 2 on the Memory Card System, when an Init or System__init is programmed. In any mode of operation, Reference Register 2 truncates; i.e., sets a limit on,the lowest memory location being read from the card.

*For Buffered Random Scanning applications, this can be the same Memory card specified as external channel address memory where:

Buffer Memory $<=$ (Total amount of Memory - Channel Address Memory)
and Buffer Memory $=$ (# of A/D cards) $*$ (# of Timebase pulses)

Menu Page 2 of 2 (continued)

Reference Count

Appears only when the Memory Type is 69790B. This specifies the value programmed into the Reference Register on the Memory card when an Init or System__Init is programmed. This value is the highest memory location that can be read from the card. The default is 4095. The function of the Reference Count depends upon the mode of operation, and the buffer direction that is in effect:

1. In FIFO mode with the IN direction, the Reference Count sets the number of words that will be stored into the Memory card by an external device before it will indicate that memory is "almost full".
2. In FIFO mode with the OUT direction, the Reference Count sets the number of words that remain in the Memory card before it will indicate that memory is "almost empty".
3. In Recirculate mode with the OUT direction, the Reference Count truncates; i.e.,sets a limit on the size of memory being read from the card by an external device.
4. In Recirculate mode with IN direction, the Reference Count is meaningless.

or

Ref Reg 1

Appears only when the Memory Type is 69791A. This specifies the value programmed into Reference Register 1 on the Memory Card System, when an Init or System__init is programmed. This value is the highest memory location that can be read from the card. The default is 65,535. The function of Reference Register 1 depends upon the mode of operation and the buffer direction that is in effect:

1. In FIFO mode with the IN direction, Reference Register 1 sets the number of words that will be stored into the Memory Card System by an external device before it will indicate that memory is "almost full".
2. In FIFO mode with the OUT direction, Reference Register 1 sets the number of words that remain in the Memory Card System before it will indicate that memory is "almost empty".
3. In Recirculate mode with the IN/OUT direction, Reference Register 1 truncates; i.e., sets an upper limit on, the size of memory being read from the card by an external device (Note that the write pointer is not affected).

Buffer Direction

Specifies whether the front-end card will be an input card or and output card. The default is IN.

Front End Type

Specifies the card that will be referenced by the Memory card or Memory Card System in the multiple card function. A Buffer Direction of IN will allow you to specify either a 69751A or 69759A A/D input card. A Buffer Direction of OUT will allow you to specify either a 69720A or 69721A D/A output card. The default is a 69751A.

Menu Page 2 of 2 (continued)

A/D Name(s)

or

D/A Name

Appears only when the Front End Type is either a 69751A/69759A A/D card. This specifies the name of the input card being referenced by the Buffer. Press the change key to step through each name listed in the COM statement and select the name of the front-end input card when it appears. In the case of the 69759A A/D card used with a 69791A Memory Card System, multiple names can be selected because the CHAIN IN and CHAIN OUT features of the 69759A A/D card allow several A/D cards to be multiplexed to one 69791A Memory Card System *. If the selected name has not yet been configured, an error will be generated. At this time, the user can continue, but the selected name **MUST** be configured before end menu. If the name has not been configured as either a 69751A or 69759A an error will be generated. There is no default. This parameter **MUST** be filled in.

Appears only when the Front End Type is either a 69720A/69721A D/A card. This specifies the name of the output card being referenced by the Buffer. Press the change key to step through each name listed in the COM statement and select the name of the front-end output card when it appears. If the selected name has not yet been configured, an error will be generated. At this time, the user can continue, but the selected name **MUST** be configured before end menu. If the name has not been configured as either a 69720A or 69721A an error will be generated. There is no default. This parameter **MUST** be filled in.

* Note that the order in which you select the names **MUST** be the same as the order in the chain. For example, the first name you select must correspond to the first card in the chain, the second name must correspond the second card in the chain, etc.

Programming Fundamentals

Only a few statements are needed to program the Buffer function. A complete example using these statements, can be found in Application Note 316-5. Further information about multiple card functions can be found in Chapter 10. The following is a description of the fundamental statements.

Statement	Description
Init(Buffer_name)	Loads the Memory card or Memory Card System with the initial values for the reference register(s), Mode, Direction, timeout, and external lockout as specified in the Buffer menu. Also, the associated front-end card is loaded with the initial values specified in that particular card's menu. If the front-end card is either a 69751A or 69759A ,with an associated scanner, the 69750A Scan Control card is loaded with the initial values specified in the associated menu. This instruction also disables interrupts from all cards in the buffer configuration. The name **MUST** be the Buffer name.
Input_rblock (Buffer_name,Array(*) [,Count[,Index]])	Reads a block of data from the Memory card or Memory Card System, converts the data according to the Standard Data type and Data Conversion specified for the front-end card and returns the data in the REAL array called "Array". The card must be in either the FIFO input or Recirculate input mode; if it is not, an error message (code 825) will be generated. With front-end 69759A A/D card(s), the range source of each A/D card must be set to internal or an error (code 846) will be generated. For multiplexed A/D applications (with the range source of every A/D card set to internal) the data read back from the Memory Card System is from every A/D involved. If three A/D cards are multiplexed, the first three readings are stored from each A/D card, then the next three readings are stored from each A/D card etc.; every third reading is from the third A/D card. To obtain data of a particular A/D card, use Input_iblock followed by Demultiplex (as described in the advanced programming statements). This also allows you to supply range values externally.

NOTE

If your computer has cache memory, you may get a BASIC error (388) when using this statement with DMA transfers. In this case, turn off cache memory before executing the statement and turn it on again after the statement. The command to turn cache memory off is:

CONTROL 32,1;0

The command to turn cache memory on is:

CONTROL 32,1;1

The number of data words read from the Memory card is equal to the size of the data array. That is, if the data array is dimensioned for 1000; then 1000 data words will be returned. Optionally, if the number of data words to be read is less than the size of the data array then a parameter, referred to as "Count", can be specified for the required number of data words. If "Count" is used, you can also use the optional parameter, "Index". Index can be used with the Count parameter to specify the position in the array where the first input data word will be placed; subsequent data words are placed in the successive array positions that follow. The data array **MUST** be previously DIMensioned or ALLOCATEd in the user's program, otherwise a BASIC system error (#13) will be generated.

Programming Fundamentals (continued)

Statement	Description
Output_rblock (Buffer_name,Array(*) [,Count[,Index]])	Converts a block of REAL data according to the front-end card's Standard Data Type and Data conversion that was selected in the menus and then sends the block to the Memory card or Memory Card System. The card must be in either the FIFO output or Recirculate output mode; if it is not, an error message (code 825) will be generated. An Output_rblock is equivalent to a Translate followed by an Output_iblock (described in advanced the programming statements). The number of words written to the Memory card is equal to the size of the data array called "Array". That is, if the data array is dimensioned for 2048 words; then 2048 words will be written to the Memory card. Optionally, if the number of data words to be sent is less than the size of the data array then the parameter, "Count", can be specified for the required number of data words that will be sent to the Memory card. The optional parameter, "Index" can be used with the Count parameter to specify the array position of the first data word that will be sent. The data array **MUST** be previously DIMensioned or ALLOCATEd in the user's program, otherwise a BASIC system error (#13) will be generated.

The operating mode of the buffer memory can be changed from that selected in the Configuration phase by using one of the following four statements.

Set_fifo_in(Buffer_name)	Changes the Memory card or Memory Card System's operating mode to the FIFO Input mode.
Set_recirc_out (Buffer_name)	Changes the Memory card or Memory Card System's operating mode to the FIFO Output mode.
Set_recirc_in (Buffer_name)	Sets the Memory card or Memory Card System's operating mode to the Recirculate Input mode.
Set_recirc_out (Buffer_name)	Sets the Memory card or Memory Card System's operating mode to the Recirculate Output mode.

Advanced Programming

There are 14752A statements that allow special purpose programming of the Buffer multiple card functions. They are considered advanced because you need a good understanding of the I/O cards and system to put together a meaningful program. With the exception of Translate and Demultiplex, these statements are the same as the advanced programming statements listed for the 69790B Memory card and the 69791A Memory Card System. Depending upon the Memory Type chosen for the application, either set of statements can be used. The Buffer name must be specified for the Name parameter. Further information about multiple card functions can be found in Chapter 10 and complete examples can be found in Application Note 316-5 on Data Capture.

The Translate and Demultiplex statements described shortly allow the user to separate the data conversion process from the measurement process. This capability is provided by combining Translate or Demultiplex with Input__iblock(for front-end input cards) or Output__iblock (for front-end output cards).

For front-end Input cards, the Input__iblock statement is used to quickly collect raw integer data from the Memory card or Memory Card System. The Translate or Demultiplex statement can be used at a later time to process this integer data into real numbers using the data conversion of the front-end A/D card. Input__iblock followed by Translate is equivalent to Input__rblock except that Translate allows the user to delay the processing of data. Input__iblock followed by Demultiplex allows the user to delay the processing of one particular A/D card for multiplexed A/D applications. Note that Input__rblock can not be used for applications involving the 69759A A/D card(s) where the range source is set to external. This is because Input__rblock does not allow the user to specify the associated range values. In this case, Input__iblock followed by Translate or Demultiplex **MUST** be used.

For Output cards, the Translate statement is used to convert real numbers into raw integer form using the data conversion of the front-end D/A card. The Output__iblock can then write this integer data to the Memory card or Memory Card System. Translate followed by Output__iblock is equivalent to Output__rblock.

Statement	Description
Translate (Front__end__name, Iarray(•),User(•) [,Range(•)])	Used in conjunction with Input__iblock or Output__ iblock, described in the 69790B and 69791A advanced programming sections,to process front-end card data.The front-end A/D input card or D/A output card is specified by the parameter, Front__end__name. The array named "Iarray" is an array of Integers. Since this is the same array as "Iarray" used in Input__iblock and Output__iblock,the array names are the same. For an input card, this is the source array for the Translate statement and for the an output card it is the destination array. The array named "User" is an array of REAL numbers. For an input card this is the destination array and for an output card this is the source array. In other words, for an input card, integer data is taken from Iarray, converted to real numbers and returned into the User array. For an output card, data is taken from the User array, converted to integer numbers and returned into Iarray.

Advanced Programming (continued)

Statement	Description
	The optional Range array parameter can be used to process the results if the front -end card is the 69759A A/D card with the range source set to external. Depending upon the full-scale range selected,the range array **MUST** contain either 10,1,.1,100 values, or 5,.5,.05,50 values. Each value in the range array will be associated with each corresponding value in the source array. If the range array is smaller than Iarray, the ranges will be reused until the Iarray is exhausted. For example, if the range array contains 5 elements and Iarray contains 15 elements, each range value will be used three times (range element #1 will be used with Iarray element #1,6 and 11, value #2 will be used with Iarray element #2,7 and 12, etc). This range array can be the same array used in the Load__ channels statement (refer to Scanner section of Chapter 5). All arrays **MUST** have been previously DIMensioned or ALLOCATEd in the user's program otherwise a BASIC system error (#13) will be generated.
Demultiplex (Buffer__name,Adc__name, Iarray(•),Destination(•) [,Range(•)])	Only used for multiplexed 69759A A/D applications. Used after Input__iblock, described in the 69791A advanced programming section, to process the data of one particular A/D card. Data of the A/D card specified by the parameter,Adc__name is retrieved from the INTEGER array named "Iarray". Since this array is the same as "Iarray" used in Input__iblock,the array names are the same. The data retrieved from Iarray is then converted to real numbers and returned into the REAL array named "Destination". The optional Range array parameter is used to process the results when the range source of the A/D card is set to external. Depending upon the full-scale range selected, the range array **MUST** contain either 10,1,.1,100 values or 5,.5,.05,50 values. Each value in the range array will be associated with each corresponding value in the source array. If the range array is smaller than Iarray, the range values will be reused until Iarray is exhausted. For example, if the range array contains 5 elements and Iarray contains 15 elements, each range value will be used three times (range value #1 will be used with Iarray element #1,6 and 11, value #2 will be used with element #2, 7 and 12, etc). This range array can be the same array used in the Load__channels statement (refer to Scanner section of Chapter 5). All arrays **MUST** have been previously DIMensioned or ALLOCATEd in the user's program otherwise a BASIC system error (#13) will be generated.

Chapter 10
Advanced Programming Concepts

This Chapter contains information that will be helpful in the development of programs for more sophisticated CAT applications using the 14752A statements and the Multiprogrammer hardware. While new users may not need this information in the beginning, it is still beneficial since it discusses some programming concepts, and it also explains how some of the hardware functions.

Multiple Naming

This section explains the concept of defining more than one Name for a single hardware function. There are two instances where it may be convenient to use multiple names:

First, when it is desirable to independently access a single bit or a group of bits in a function. The functions that allow grouping of bits have the parameters Start Bit and Length in their menus.

Second, when the configuration (such as mode, period, data conversion, etc.) of a function must be changed from one state to another within the application program.

When using more than one Name for a single function, keep the following in mind:

1. All Names **MUST** be initialized, by using either the Init or System__init statements, before programming any other 14752A statements that reference the Name. If you fail to do so, an error message (Code 815) will be generated when you try to program the Named function.

2. If multiple (Configured) Names are in the COM statement and you issue a System__init statement, they will all be initialized in the order that they appear in the COM statement. Naturally, only the last configuration will remain standing (active). If this is not acceptable, you can use individual Init statements instead or you can reorder the names within the COM statement.

3. If more than one Name is used for a single function, you **MUST** initialize each Name **every time** before using it, with the exception of the following functions: 69730A, 69731B, 69751A, 69759A, 69770A, 69771A, 69775A, and the 3478A DMM; these functions don't need to be re-initialized in order to "switch" Names.

4. There are certain considerations that apply to interrupts from multiple Named functions discussed later under "How to Handle Interrupts". Briefly stated here: interrupts must be re-enabled (with an Enable__intr statement) after an Init or System__init statement is programmed, and the Reason that is returned is that of the most recently enabled Name.

Multiple Names for Functions That Allow Grouping of Bits

There are several Multiprogrammer input and output (I/O) cards that allow grouping of bits:

69730A Relay Output
69731B Digital Output
69770A Isolated Digital Input
69771A Digital Input / Analog Comparator

These cards all have individual bits at their external edge connector. (The 69730A's relays can be considered bits since each relay is programmed open or closed by a single bit.) The 14752A CAT Programming Package allows multiple Names to be defined for each of these cards, dividing their 16 bits into groups of one or more bits. The groups can be from 1 to 16 bits in width, but the bits in a group must be contiguous. The Start Bit and Length menu parameters are used to define the bits associated with any particular group. Refer to Chapters 7 and 8 for examples and more information on Start Bit and Length.

All 16 bits of the I/O function are read from, or written to, using a single 16-bit word. However, for output cards (69730A and 69731B), the 14752A subprograms will merge new data for a particular group into the 16 bit word that is sent to the card. For the input cards (69770A and 69771A), the subprograms read the full 16 bit word from the card, mask off the bits which are not related to those defined for the group, and then return a decimal value represented by only those bits. If you plan to use the Start statement with these cards, you should read the section on the next page entitled "Synchronization of Program and I/O"

Naming Multiple Mode Functions

Many 14752A supported functions have more than one mode of operation. For example, the 3478A Digital Multimeter can make voltage, current, or resistance measurements, the 69790B Memory card and 69791A Memory Card System can operate in a FIFO mode or a recirculate mode, and the 69759A A/D card can be programmed to take measurements using any one of four actual voltage ranges. When different modes of a function are to be used in a single program, it is convenient to give each configuration, or set of modes, its own Name. Thus, during program execution, a new set of parameters can be selected by using a different Name for the function. Each I/O function, however, requires you to "Init" the second Name after you have been using the first Name to control the function; the 69790B Memory card/69791A Memory Card System is such a function, but the 3478A can be controlled by Multiple Names without using the Init statement each time.

For an application showing where multiple configurations can be useful, lets assume that we want to measure the characteristics of a signal in terms of DC and AC magnitude. We can use two different Names for the 3478A Multimeter. We can call one Name "Level" and use its configuration menu to set up the 3478A to function in the DC mode. We can call the other name "Ripple" and use its menu to set up the 3478A to function in AC mode. An example program using these Names follows:

```
10 COM/Names/Level,Ripple
20 REAL Volts_dc,Volts_ac
30 Configure ("Menu_on","ACDC")
40 System_init
50 Input (Level,Volts_dc)
60 Input (Ripple,Volts_ac)
70 PRINT Volts_dc,Volts_ac
80 END
```

Synchronization of Program and I/O

There are times when you may want to synchronize your program execution with actual input and output (I/O) events at the card edge. At other times, once you have programmed a function, you may not care when it actually finishes until some time later in your program. Both of these cases can be handled with the use of a few advanced programming statements described in the following paragraphs. Note that all of these statements, except Wait_for, complete execution very quickly, and therefore, will not suspend program execution.

Output Programming Using Preset, Start, Wait_for, and Check_done

Multiprogrammer output cards employ a data holding register to allow loading of data before transfering it to the card edge outputs. This data holding register is called First Rank Storage and the output registers are called Second Rank Storage. The fundamental statement, Output, loads data into the first rank register, transfers the data to second rank, and waits for the operation to complete all in one step. This allows card outputs to be changed sequentially, one card after another, but does not allow simultaneous outputs from multiple cards in parallel. Card outputs can be changed simultaneously by first loading data into each card's First Rank Storage with Preset statements, and then transferring each card's data to Second Rank with a Start statement or External Trigger signal. This process is illustrated in Figure 10-1 and the actions are individually explained in the following paragraphs.

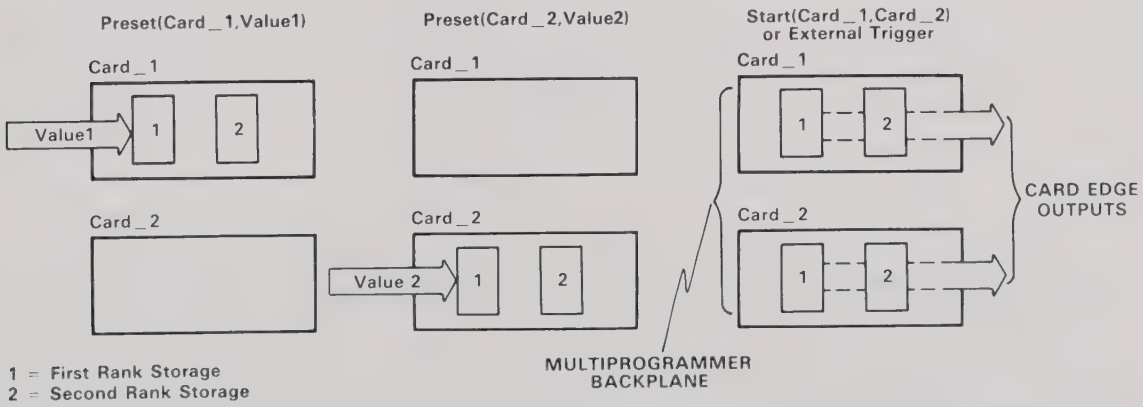


Figure 10-1. Output Programming Using Preset And Start Statements.

The Preset Statement. The Preset statement loads data into the First Rank Storage registers on output cards. This data does not cause the output at the card edge connector to change until either a Start statement or an External Trigger signal transfers the data to Second Rank Storage.

The Start Statement. The Start statement causes data held in First Rank Storage to be loaded into Second Rank Storage, thereby generating the card's outputs at the edge connector.

The Start statement has two primary applications when used in combination with the Preset statement for output operations:

1. Preset and Start will simulate an Output, except that you will **NOT** wait for the operation to be completed.

2. Start allows up to 8 I/O functions (Inputs or Outputs) to be simultaneously initiated under program control. All of the Names in a Start statement must be located at the same 98633A Multiprogrammer Interface select code, since it uses the group address flag (GAFF) capability of the 697XX-series I/O functions to "cycle", or initiate a read on, all the I/O cards at the same time or all cards must be located at the same HP-IB address (i.e. in a single 6942A and/or any 6943 extender frame connected to it.)

If you have assigned multiple Names to a Multiprogrammer function that allows grouping of bits, you must remember that, even though you treat them as independent Names, they are all part of the same I/O card. Therefore, when more than one Name is assigned to a single 69730A, 69731B, 69770A, or 69771A card, and one of the Names is Started, then all the Names associated with the card are Started. For example, let's use a 69731B Digital Output card with bits 0 through 7 assigned the Name Out__1, and bits 8 through 15 the Name Out__2. We can use two Preset statements to send Out__1 the number 25 and Out__2 the number 10. Now to get the data from First Rank Storage to the edge connector we can use a single Start statement like this:

```
Start (Out__1)
```

This statement will start **BOTH** Names simultaneously. Since either Name will start the card, either one could have been used. If we really wanted the two Names to start sequentially, we could have used either two Output statements or two Start statements like this:

```
Output (Out__1,25)
Output (Out__2,10)
        or
Preset (Out__1,25)
Start (Out__1)
Preset (Out__2,10)
Start (Out__2)
```

The only difference between these two methods is that the Output statements will wait for the card to finish and the Preset statements will not wait. As before, either Name could have been used to Start the card.

The External Trigger Signal. Receipt of an External Trigger signal has the same effect as the Start statement, but allows the action to be synchronized by an external event. An External Trigger signal can be connected to cards in any frame, and multiple cards can be wired together. The polarity of the External Trigger signal can be changed via another signal called Trigger Polarity Select. For details on these and other signals, see the "Frontplane Signal and Timing" section in this chapter. The 69790B Memory card/69791A Memory Card System and the 69750A Scan Control/Pacer card have external trigger lockout capability, and many output cards (eg. 69720A, 69721A, 69735A, 69736A) have an external trigger bypass switch which will decrease the delay time between an External Trigger signal and the card's output. (Refer to the appropriate I/O card Operating Manual.)

The Check__done and Wait__for Statements. Both the Check__done and Wait__for statements will indicate to the program whether the functions have completed their output operations or not. After the execution of either the Start statement or an External Trigger, a period of time elapses before a function completes its operation. This time is dependent on the function and in some cases on the parameters of the operation. For example, a Pulse Output card does not complete its operation until

the specified number of pulses has been output. A Digital Output card does not complete its operation until the Gate/Flag handshake has been completed (see the "Frontplane Signals and Timing" section for a description of the Gate/Flag handshake).

The simplest and fastest way for the program to find out if a function has completed its operation is to use the Check_done statement. The Check_done statement returns a Flag, "1" or "0", which reflects the state of the operation (1 if the operation is complete; 0 if it is not). Once the Check_done Flag is returned as a 1, Check_done returns a 0 until the function completes again. The Check_done statement does not suspend program execution. Here's the programming sequence using the Check_done statement to determine when the function called "Dig_out" is finished:

```
.
.
40 Preset(Dig_out,10)
50 Start (Dig_out)
60 Loop: Check_done(Dig_out,Flag)
70 IF Flag=0 THEN Loop
80 PRINT "Dig_out function is finished."
.
.
```

Any statement that can timeout will perform an operation similar to a Check_done statement. If you execute a Check_done statement after any of these statements, Flag will be set to zero. Therefore, use Check_done only in conjunction with a Start statement or External Trigger input signal. The Wait_for statement is used to suspend the execution of the program until a function has completed its operation after it has been started with a Start statement or external trigger signal. The Wait_for statement checks the status of a card's operation and waits until it has completed. This process can continue until the Timeout value specified in the menu has elapsed. If the function does not complete before the expiration of the timeout, an error message (code 803) is generated. The default value for Timeout is 5 seconds and can be changed either in the configuration menu or by using the Set_timeout statement.

The Wait_for and Check_done statements are particularly useful if you have an application which utilizes the External Trigger signal. Once data has been loaded into a card with the Preset statement, an external device can signal the card to generate its output with the External Trigger control line. Since this signal is generated by an external device and not by the program, it is necessary to have the card inform the program when it has been triggered. The Wait_for or Check_done statement can be used to determine when the card has finished its output.

The occurrence of function complete, as signalled by the End Of Process line going high, can also be used to generate an interrupt to the computer. The End of Process line (EOP) is described later in this chapter under "Frontplane Signals and Timing." This method is particularly useful if external triggers occur truly asynchronously and not in response to other known I/O operations. Information about using interrupts is given later in this Chapter.

Input Programming Using Start, Wait_for, Check_done, and Get

Multiprogrammer input cards provide a data holding register, called the Input Latch, to allow simultaneous acquisition from several input cards in parallel. The fundamental programming statement, Input, strobes a single card, waits for the operation to complete, and reads the acquired data into a variable all in one step. This allows cards to acquire data sequentially, but does not allow multiple input cards to acquire data simultaneously.

Input cards can acquire data simultaneously by first triggering each card with a Start statement or the External Trigger input signal. Then the Check__done and Wait__for statements can be used to check whether the functions have completed or not. When the functions complete, they can be read one at a time using Get statements. This process is illustrated by Figure 10-2 and the actions are individually explained in the following paragraphs.

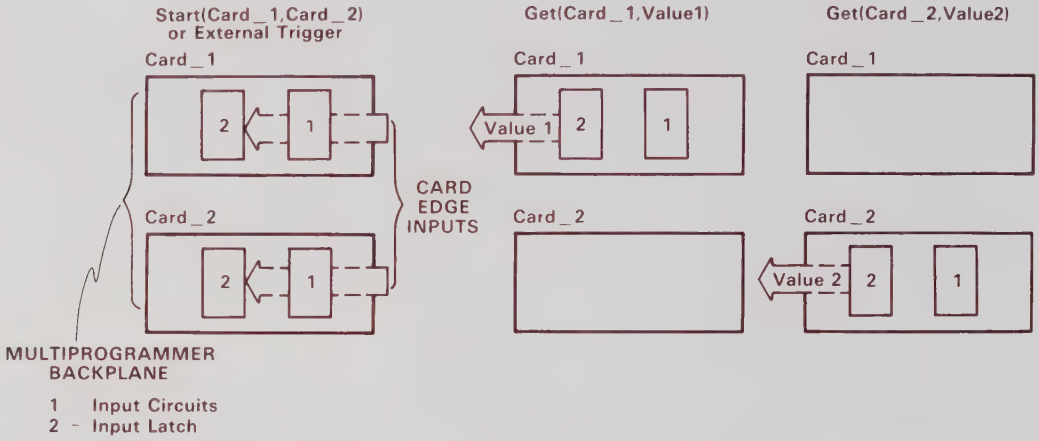


Figure 10-2. Input Programming Using Start And Get Statements.

The Start Statement. The Start statement causes input cards to acquire data from their external edge connectors under program control.

The Start statement has two primary applications for input operations when used in combination with a Wait__for (or Check__done) statement and a Get statement:

1. Start initiates an input operation without requiring you to wait until the operation completes. It allows you to come back at a later time to read back the data.

2. Start allows up to 8 I/O functions (Inputs or Outputs) to be simultaneously initiated. All of the Names in a Start statement must be located at the same 98633A Multiprogrammer Interface select code, since it uses the group address flag (GAFF) capability of the 697XX-series I/O functions to "cycle" all the I/O cards at the same time or all cards must be located at the same HP-IB address (i.e. in a single 6942A and/or any 6943A extender frame connected to it).

If you have assigned multiple Names to a Multiprogrammer input function that allows grouping of bits, you must remember that, even though you treat them as independent Names, they are all part of the same I/O card. This is analogous to the situation explained earlier for output functions. That is, when you Start one name that is associated with an input card that supports bit groupings, you Start all the Names on the card, even if they were not included in the Start statement. When you have determined that the input operation is complete, you can execute a Get statement for each Name which is associated with the card, and read back the data that was latched by the Start statement.

The External Trigger Signal. The External Trigger signal has the same effect as the Start statement, but allows the action to be synchronized by an external event. An External Trigger signal can be connected to cards in any Multiprogrammer frame or to the 3478A Digital Multimeter. The External Trigger input of multiple functions can also be wired together. The polarity of the External Trigger signal can be changed via another signal called Trigger Polarity Select. For details on these and other

signals, see the "Frontplane Signal and Timing" section in this chapter. The 69790B Memory card/69791A Memory Card System and the 69750A Scan Control/Pacer card have external trigger lockout capability and many input cards (eg. 69751A, 69770A) have an external trigger bypass switch which will decrease the delay time between an External Trigger signal and the card's input (refer to the appropriate I/O card Operating Manual).

The Check_done and Wait_for Statements. Both the Check_done and Wait_for statements will indicate to the program whether the functions have completed their input operation or not. After the execution of either the Start statement or an External Trigger, a period of time elapses before a function completes its operation. This time is dependent on the function and in some cases on the parameters of the operation. For example, a 69751A card does not complete its operation until the analog to digital conversion has been completed. A Digital Input card does not complete its operation until the Gate/Flag handshake has been completed (see the "Frontplane Signals and Timing" section for a description of the Gate/Flag handshake).

The simplest and fastest way for the program to find out if a function has completed its operation is to use the Check_done statement. The Check_done statement returns a Flag, "1" or "0", which reflects the state of the operation (1 if the operation is complete; 0 if it is not). Once the Check_done Flag is returned as a 1, Check_done returns a 0 until the function completes again. The Check_done statement does not suspend program execution. Here's the programming sequence using the Check_done statement for input programming:

```
.
.
.
50 Start(Dig_in)
60 Loop: Check_done(Dig_in,Flag)
70 IF Flag=0 THEN Loop
80 Get(Dig_in,Data)
90 Print "The data from Dig_in is"; Data
.
.
.
```

Any statement that can timeout will perform an operation similar to a Check_done statement. If you execute a Check_done statement after any of these statements, Flag will be set to zero. Therefore, use Check_done only in conjunction with a Start statement or External Trigger input signal.

The Wait_for statement is used to suspend the execution of the program until a function has completed its operation after it has been started with a Start statement or external trigger signal. The Wait_for statement checks the status of the card's operation and waits until it has completed. This process can continue until the Timeout value specified in the menu has elapsed. If the function does not complete before the expiration of the timeout, an error message (code 803) is generated. The default value for the Timeout is 5 seconds and can be changed in the menus or with a Set_timeout statement.

The Wait_for and Check_done statements are particularly useful if you have an application which utilizes the External Trigger signal. Once data has been loaded into a card by an external device with an External Trigger signal, it is necessary to have the card inform the program when it has been triggered so you can read the data from the card with a Get statement. The Wait_for or Check_done statement can be used to determine when the card has finished its acquisition.

The occurrence of function complete (signalled by the End Of Process line going high) can also be used to generate an interrupt to the computer. This method is particularly useful if external triggers occur truly asynchronously and not in response to other known operations. Information about using interrupts is given later in this Chapter.

The Get Statement. The Get statement is used to read data from the Input Latch after the card has completed an operation initiated by the Start statement or an External Trigger input signal.

Note that the 3478A Digital Multimeter does not have dual rank storage. When a reading is requested from the 3478A using the Get statement, the program will wait until a reading is available. If none becomes available within the 5 seconds allotted for HP-IB Bus Timeout (this is **NOT** the 3478A Timeout you specified in the menus), an error message is generated.

How to Handle Interrupts

You have already seen two ways to determine when a function completes its operation; recapped here they are:

1. Using a fundamental statement that can timeout; e.g., Output, Input.

2. Using Check_done or Wait_for statements after the function was started by a Start statement or External Trigger signal.

This section will show you a third way, using interrupts, to determine when a function has completed its operation. Like the Check_done and Wait_for statements, interrupts **MUST** be used only with a function that has been started by a Start statement or External Trigger signal. Also, after enabling interrupts from a function, you must never use a Check_done statement or any statement that will timeout; e.g., Output, Input, or Wait_for.

Interrupts are useful when the program needs to respond to a hardware event which occurs asynchronously with program events. It may be time consuming or impossible to sample the function complete condition with the Check_done statement or suspend program execution with the Wait_for statement. The 14752A CAT Programming Package supplies two easy methods of dealing with interrupts. One way to handle interrupts is to use a subprogram called "User_handler" that is described later in this chapter. The other way to handle interrupts is to write your own subroutine as described in the paragraphs following the "User_handler" description.The overhead associated with servicing an interrupt using subprograms is approximately 13.7 milliseconds. An alternate approach of writing your own subroutine reduces the overhead to approximately 4 milliseconds at the expense of several additional lines of code and **context-sensitive interrupt linkage.** If an interrupt occurs while a BASIC program line is executing, the interrupt is not serviced until after the BASIC program line is finished. For this reason the interrupt service time is measured from the time the BASIC line finishes (EOL) to the time that the interrupt service routine is entered. For example, if an interrupt condition occurs just after a WAIT 10 statement begins execution, the interrupt service routine will not be entered for at least 10 seconds. When speed is a concern, it is worthwhile to note that a Check_done statement will take as little as 1.5 milliseconds to execute, including the BASIC "IF...THEN..." statement, as opposed to the longer time required to process an interrupt. The Check_done statement, therefore, represents the fastest means of reaching a service routine from the time a function completes.

The servicing of interrupts just described is only one type of interrupt servicing that occurs in your computer. The servicing of "ON ERROR", "ON END", and "ON TIMEOUT" statements command the highest priority in the system. Next in the interrupt hierarchy is the interrupt servicing that takes place in your program as a result of your "ON INTR", "ON CYCLE", and "ON KEY", etc. statements. These statements can be assigned a priority from 0 to 15 (15 being the highest). The default priority for these statements is 1. The default results from a statement such as:

```
ON INTR 29 CALL Intr_handler
ON INTR  8 CALL Intr_handler
```

"29" is the default select code of the 98633A Multiprogrammer Interface and "8" is the default select code of the 98624A HP-IB Interface used with a 3478A or 6942A. It is recommended that you assign an interrupt priority of "3". To assign a priority of 3 to the same interrupt processing, use this statement:

```
ON INTR 29,3 CALL Intr_handler
ON INTR  8,3 CALL Intr_handler
```

NOTE

If BASIC error 163 occurs while using the ON INTR or OFF INTR statements with the 98633A interface, you have not loaded the binary as described in Chapter 3.

Finally, the Multiprogrammer itself assigns an interrupt priority to the cards within the frames in the link. The card slots can be numbered from 0 to 127 (the slot within the frame plus sixteen times the frame number). The card with the lowest numbered card slot has the highest priority. The highest numbered card slot has the lowest priority.

Using the 14752A Subprogram "Intr_handler"

In response to the occurrence of an interrupt, the following tasks must be performed:

1. Identify the interrupt device.
2. Service the interrupting device.
3. Re-enable the interrupting select code.
4. Resume program execution starting fron where it was interrupted.

Operations 1 and 4 above are handled by the subprogram "Intr_handler" provided with the 14752A. Operations 2 and 3 are accomplished by a subprogram written by you called "User_handler", which is automatically called by Intr_handler. Note that, User_handler **MUST** be the label for your interrupt service routine when using the Intr_handler subprogram. The flowchart in Figure 10-3 illustrates the sequence of events when an interrupt occurs:

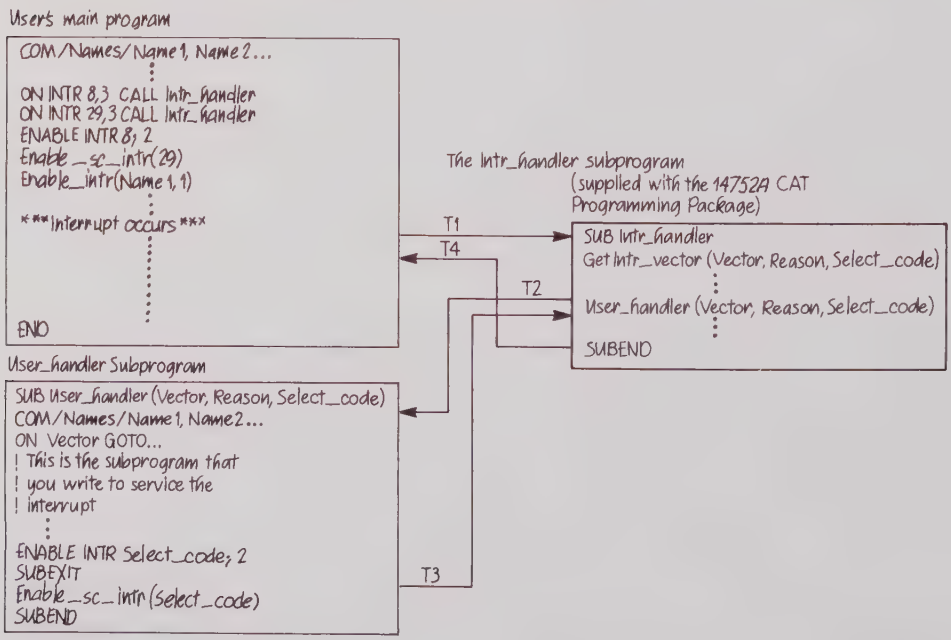


Figure 10-3. Interrupt Handling With A Subprogram

From this flowchart you can construct a program to handle interrupts. On the next page is a program skelton that illustrates the basic structure that is needed for servicing interrupts with a subprogram. Following it there is a brief explanation of the pertinent program lines.

```
10    !-----------------------------------------------------------------
15    !   Example program using subprogram for interrupt service routine
20    !-----------------------------------------------------------------
25    !
30    COM /Names/ Dvm                 ! HP3478A on select code 8
35    COM /Names/ Valve,Flow_meter    ! I/O cards in a HP6942A on select code 8
40    COM /Names/ Dac,Relay,Digout    ! I/O cards in a HP6944A on select code 29
45    !
50    Configure("ASK ME","MY_CONFIG")
55    !
60    System_init
65    !
70    !---------------- Set up interrupt linkages ------------------------
75    !
80    ! Set up linkages for handling interrupts from the HP-IB interface
85    ! on select code 8 and the 98633A interface on select code 29.
90    ! The subprogram "Intr_handler" is part of the 'LIBRARY' subprogram.
95    !
100   ON INTR 8,3 CALL Intr_handler
105   ON INTR 29,3 CALL Intr_handler
110   !
115   ! The following statement allows the interrupt system to respond to
120   ! service request (SRQ) from the HP-IB interface
125   ! (HP3478A or HP6942A interrupt).
130   !
135   ENABLE INTR 8;2
140   !
145   ! The following statement allows the interrupt system to resopnd to
150   ! interrupt from the HP98633A interface (HP6944A interrupt).
155   !
160   Enable_sc_intr(29)
165   !
170   ! The following statements enable the interrupt logic of the
175   ! individual I/O function to generate backplane interrupt.
180   ! It also defines the value to identify the interrupting function.
185   !
190   Enable_intr(Valve,1)
195   Enable_intr(Dvm,2)
200   Enable_intr(Relay,3)
205   !
210   !---------------- Body of application program ----------------------
215   !
220   PRINT "Start of program."
225   !
230 Wait4ever: GOTO Wait4ever
235   !
240   ! The body of your application program goes here
245   !
250   PRINT "End of program."
255   !
260   END
265   !
270   !---------------- Interrupt service subprogram ---------------------
275   !
280   ! This program is called automatically by "Intr_handler"
285   ! whenever an interrupt on select code 8 or select code 29 occurs.
290   !
295   SUB User_handler(Vector,Reason,Select_code)
300   !
305   ! The /Names/ common block declaration must
```

```
310   ! be identical to that in the main program.
315   !
320   COM /Names/ Dvm,Valve,Flow_meter,Dac,Relay,Digout
325   !
330   ! Based on "Vector" (the number programmed in the "Enable_intr"
335   ! statement).  Perform the appropriate interrupt service routine.
340   !
345   ON ERROR GOTO Bad_intr
350   ON Vector GOTO Valve_svc,Dvm_svc,Relay_svc
355   OFF ERROR
360   !
365 Valve_svc:    ! Service routine for 'Valve'
370               PRINT "Valve interrupted."
375               GOTO Exit_hpib_svc
380   !
385 Dvm_svc:      ! Service routine for 'Dvm'
390               PRINT "Dvm interrupted because ";
395               IF BIT(Reason,0) THEN Data_ready
400               IF BIT(Reason,1) THEN Frontpanel_srq
405               GOTO Bad_intr
410   !
415 Data_ready:   PRINT "it took a reading."
420               GOTO Exit_hpib_svc
425   !
430 Frontpanel_srq: PRINT "the SRQ button was pushed."
435                 GOTO Exit_hpib_svc
440   !
445 Relay_svc:    ! Service routine for 'Relay'
450               PRINT "Relay interrupted."
455               GOTO Exit_iofc_svc
460   !
465   ! Handler for bad interrupt
470   !
475   ! Bad interrupt can be a vector, a reason or an interrupt
480   ! from a select code which you did not expect occured.
485   !
490 Bad_intr:     !
495               PRINT "Unexpected interrupt at select code ";Select_code
500               PRINT "Vector = ";Vector;"    Reason = ";Reason
505               STOP
510   !
515   !---------------- Interrupt service routine exit --------------------
520   !
525 Exit_hpib_svc:  !
530   !
535   ! Further interrupts from the HP-IB "Select_code" are disabled while
540   ! while in "User_handler".  If you wish to allow additional interrupts
545   ! to occur, they must be expressly re-enabled by you.
550   !
555   ENABLE INTR Select_code;2
560   SUBEXIT
565   !
570 Exit_iofc_svc:  !
575   !
580   ! Further interrupts from the HP98633 interface at "select_code" are
585   ! disabled while in "User_handler".  If you wish to allow additional
590   ! interrupts to occur, they must be expressly re-enabled by you.
595   !
600   Enable_sc_intr(Select_code)
605   SUBEND
```

You must include the BASIC statement "ON INTR" (lines 100-105) along with the BASIC statement "ENABLE INTR" (lines 135) for the 3478A's, 6942A's and the 14752A statement "Enable_sc_intr (line 160) for the 6944A's, in your program. This must be included for each select code used in the menus. The optional parameter after the select code number in the "ON INTR" statement specifies the interrupt priority. Here, the interrupt priority has been set 3, instead of the default value of 1. This keeps interrupts from the keyboard and the real time clock form interrupting the instrument service routines. The "ON INTR" statement transfers program control to the Intr_handler subprogram when an interrupt on the specified select code occurs. This subprogram is provided as part of the 14752A CAT Programming Package.

For the 3478A, the "ENABLE INTR" (line 135) statement must be executed to allow the 98624A HP-IB interface to interrupt the computer. The "8" is the default select code assignment of this interface. The "2" following the select code is needed to tell the interface that instrument service request (SRQ's) are the conditions on which interrupts to the computer should be generated. For the 6944A, the "enable_sc_intr (line 160) statement must be executed to allow the 98633A Multiprogrammer Interface to interrupt the computer. The "29" is the default select code assignment of this interface.

Before any function can request service, it must be enabled by using the Enable_intr statement. The Enable_intr statement also associates a "Vector" with the Name of the function. "Vector" is a number provided by the user to identify the interrupting I/O function. The Enable_intr statement must come after any Init or System_init statements since these statements disable I/O function interrupts. The "Disable_intr" statement is used to prevent a function from interrupting. In the program skeleton the function "Valve" is only able to cause an interrupt after line 190 has been executed.

When an I/O function interrupts, the 14752A subprogram Intr_handler is called. The Intr_handler subprogram determines which function caused the interrupt, and extracts the interrupt Vector from the function. If the function can interrupt for more than one reason, the value "Reason" is also determined; if not, Reason is undefined. The Intr_handler subprogram then makes this information available to you by calling your subprogram which **must** be labeled User_handler (line 295).

Interrupt Handling User the Subprogram "User_handler"

Once an interrupt occurs and Intr_handler is called, program control is passed to the User_handler subprogram. User_handler is a subprogram, written by you, that will perform whatever task you want done in response to the interrupt. Since User_handler is a subprogram, it is positioned **after** the "END" statement in the main program. You should enter your User_handler (or modify the one in the EXAMPLE application program) **BEFORE** loading the LIBRARY.

The subprogram, User_handlers, is passed three variables from Intr_handler. There are: Vector Reason, and Select_code. The User_handler subprogram uses the variable Vector and Reason to pass control to the specific routine that services the interrupt. The value of Vector depends on which function caused the interrupt. In the program skeleton, Vector is 1 if the Valve function interrupts and 3 if the Relay function interrupts. Some I/O functions (69709A, 69750A, 69776A, and 3478A) can have more than one reason for interrupting the computer. The variable "Reason" is assigned a binary weighted, 16 bit 2's complement value so that you can determine the specific reason for the interrupt even if more than one reason is true. Refer to Chapter 5 for the possible values of Reason for the 69750A and 3478A. Refer to Chapter 6 for the value of Reason for the 69709A. Refer to Chapter 9 for the value of Reason for the 69776A. The last variable, "Select_code", is the select code of the interrupting 98633A Multiprogrammer Interface or the 98624A HP-IB interface.

User_handler must have its own "COM/Names/..." statement(s) (line 320). The "COM/Names/..." (line 30-40) statement(s) in User_handler must be identical to the "COM/Names/..." statement(s) in the main program. The fastest way of branching to the interrupt service code for the particular function that interrupted is via on "ON Vector GOTO label1, label2..." statement (line 350). If Vector = 1, the program will branch to label1; if Vector = 2, the program will branch to label2, etc.

In the skeleton program, if Vector is "2" at line 350, the program banched to Dvm_svc (line 385). The variable Reason uses its individual bits to represent the different conditions that could cause an interrupt. The bits are set to 1 if the corresponding condition caused an interrupt, and 0 if it did not. Note that there could be more than one cause for an interrupt, hence the bit-weighting. Reason in function-specific, and is only defined for those functions which can have multiple interrupt causes. You use Reason once you have branched to the code for the specific device that interrupted. Since the 3478A DMM is the only device which returns a valid Reason in this program, we must determine which bit of the variable, Reason, is set. The BIT function (lines 345-400) is used to test for the various interrupt conditions, in the desired priority.

Once the function - specific interrupt service statements have been executed, the program branches to "Done" or "DVM-done" (for the 3478). In order to allow further interrupts from the 3478A "Select Code" to be generated, you must execute the ENABLE INTR statement (line 555). In order to allow further interrupts from the Multiprogrammer "Select Code" to be generated, you must execute the Enable_sc_intr statement (line 600). It is convenient in many applications to re-enable them here, before the SUBEND or SUBEXIT statement for User_handler.

When the User_handler subprogram completes, control is returned to the Intr_handler subprogram. The Intr_handler then returns program control to the point where the interrupt occurred.

Interrupt Handling Using a Subroutine

As an alternative to the above method of handling interrupts, you can write a subroutine to perform all the necessary tasks. As mentioned earlier, an interrupt service subroutine will run faster than using the Intr_handler subprogram supplied with the 14752A CAT Programming Package. The key words here are "subprogram" and "subroutine". A subprogram uses a CALL statement, while a subroutine uses a GOSUB statement. The use of a "CALL" results in a longer execution time than if a "GOSUB" is used in the "ON INTR" statement.

The interrupt service subroutine referenced in the "ON INTR 8,3 GOSUB label" and "ON INTR 29,3 GOSUB label" statements must be located within the same (main program) context, i.e. **before** the END statement, and its last program line **must** be a RETURN (not a SUBEXIT or SUBEND).

NOTE

There is another issue to be considered besides interrupt service response time, and that is whether your ON INTR linkage must be context-independent or not. The linkage set up by an "ON INTR 8,3 CALL subprogram" or "ON INTR 29,3 CALL subprogram" is active regardless of whether you are executing in the main program, or in some subprogram that is called from the main program (including nested subprograms). But the linkage set up by an ON INTR GOSUB statement is context-sensitive; that is, if your main program calls a subprogram, the context changes, and the existing interrupt linkage will be deactivated. Unless your subprogram has its own ON INTR and ENABLE INTR or Enable_sc_intr statements, no interrupts will be processed while you are executing the subprogram. When you exit your subprogram and return to the main context, the interrupt linkage will be re-established and any pending interrupts are serviced at that time.

Figure 10-4 is the flow diagram showing the program flow after an interrupt where the user's subroutine (My_handler) does the servicing.

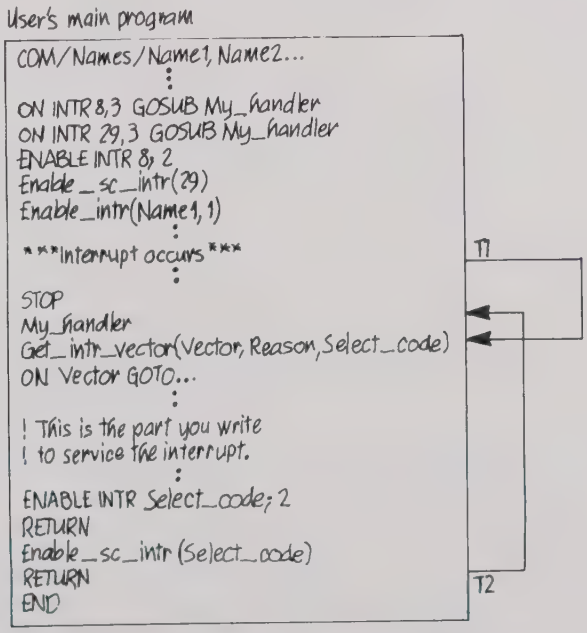


Figure 10-4. Interrupt Handling With A Subroutine

The program skeleton on the following page is very similar to the one that was given previously but illustrates the use of the "GOSUB" statement with the "ON INTR" statement. An explanation follows concerning the differences between the two methods of handling interrupts.

```
100   !-------------------------------------------------------------------------
105   !   Example program using subroutine for interrupt service routine
110   !-------------------------------------------------------------------------
115   !
120   COM /Names/ Dvm                    ! HP3478A on select code 8
125   COM /Names/ Valve,Flow_meter       ! I/O cards in a HP6942A on select code 8
130   COM /Names/ Dac,Relay,Digout       ! I/O cards in a HP6944A on select code 29
135   !
140   Configure("ASK ME","MY_CONFIG")
145   !
150   System_init
155   !
160   !----------------- Set up interrupt linkages -------------------------
165   !
170   ! Set up linkages for handling interrupts from the HP-IB interface
175   ! on select code 8 and the 98633A interface on select code 29.
180   ! The users subroutine "My_handler" must be within the same context.
185   !
190   ON INTR 8,3 GOSUB My_handler
195   ON INTR 29,3 GOSUB My_handler
200   !
205   ! The following statement allows the interrupt to respond to
210   ! service request from the HP-IB interface
215   ! (HP3478A or HP6942A interrupt).
220   !
225   ENABLE INTR 8;2
230   !
235   ! The following statement allows the interrupt system to respond to
240   ! interrupt from the HP98633A interface (HP6944A interrupt).
245   !
250   Enable_sc_intr(29)
255   !
260   ! The following statement will enable the interrupt logic of the
265   ! individual I/O function to generate backplane interrupt.
270   ! It also defines the value to identify the interrupting function.
275   !
280   Enable_intr(Valve,1)
285   Enable_intr(Dvm,2)
290   Enable_intr(Relay,3)
295   !
300   !---------------- Body of application program -------------------------
305   !
310   PRINT "Start of program."
315   !
320 Wait4ever: GOTO Wait4ever
325   !
330   ! The body of your application program goes here
335   !
340   PRINT "End of program."
345   STOP
350   !
355   !---------------- Interrupt service subroutine -----------------------
360   !
365 My_handler: !
370   !
375   Get_intr_vector(Vector,Reason,Select_code)
380   !
385   ! Make sure that the interrupting device and card
390   ! were found before servicing (select_code>0).
395   !
400   IF Select_code>0 THEN
405   !
410   ! Based on "Vector" (the number programmed in the "Enable_intr"
415   ! statement).  Perform the appropriate interrupt service routine.
420   !
425   ON ERROR GOTO Bad_intr
430   ON Vector GOTO Valve_svc,Dvm_svc,Relay_svc
435   OFF ERROR
440   !
445 Valve_svc:    ! Service routine for 'Valve'
450               PRINT "Valve interrupted."
455               GOTO Exit_hpib_svc
460   !
```

```
465 Dvm_svc:      ! Service routine for 'Dvm'
470               PRINT "Dvm interrupted because ";
475               IF BIT(Reason,0) THEN Data_ready
480               IF BIT(Reason,1) THEN Frontpanel_srq
485               GOTO Bad_intr
490   !
495 Data_ready:   PRINT "it took a reading."
500               GOTO Exit_hpib_svc
505   !
510 Frontpanel_srq: PRINT "the SRQ button was pushed."
515                 GOTO Exit_hpib_svc
520   !
525 Relay_svc:    ! Service routine for 'Relay'
530               PRINT "Relay interrupted."
535               GOTO Exit_iofc_svc
540   !
545   ! Handler for bad interrupt
550   !
555   ! Bad interrupt can be a vector, a reason or an interrupt
560   ! from a select code which you did not expect occured.
565   !
570 Bad_intr:     !
575               PRINT "Unexpected interrupt at select code ";Select_code
580               PRINT "Vector = ";Vector;"    Reason = ";Reason
585               STOP
590   !
595   !---------------- Interrupt service routine exit --------------------
600   !
605 Exit_hpib_svc:  !
610   !
615   ! Further interrupts from the HP-IB "Select_code" are disabled.
620   ! If you wish to allow additional interrupts to occur,
625   ! they must be expressly re-enabled by you.
630   !
635   ENABLE INTR Select_code;2
640   RETURN
645   !
650 Exit_iofc_svc:  !
655   !
660   ! Further interrupts from the HP98633 interface "select_code" are
665   ! disabled.  If you wish to allow additional interrupts to occur,
670   ! they must be expressly re-enabled by you.
675   !
680   Enable_sc_intr(Select_code)
685   RETURN
690   !
695   ELSE
700   !
705   ! Interrupting card not found (Select_code<0).
710   ! Determine the interface type at the select code,
715   ! re-enable the interrupt on the select code and exit.
720   ! (The HP98633A interface ID is 29)
725   !
730     IF Select_code<0 THEN
735        IF READIO(-Select_code,1)=29 THEN
740           Enable_sc_intr(-Select_code)  ! re-enable the HP98633A interface
745        ELSE
750           ENABLE INTR -Select_code;2    ! re-enable the HP98624A interface
755        END IF
760        RETURN
765   !
770     ELSE
775       !
780       ! Select_code=0 indicates an interrupt has occured
785       ! but no interrupting device was found.
790       !
795       PRINT "Interrupting device not found."
800       STOP
805     END IF
810   END IF
815   !
820   END
```

The main difference here is that the Intr_handler subprogram is not used. The Intr_handler subprogram did two things that must now be included in the user supplied subroutine, My_handler. First it defined the three parameters Vector, Reason, and Select_code. Second it checked that the value of Select_code was valid and then took the appropriate actions based on that value. In the following paragraphs, note that "My_handler" is used only for illustration. You may call this subroutine anything you choose, as long as you are consistent throughout your program; e.g., in both line 190, 195 and 365 of the example.

After the label "My_handler", the Get_intr_vector statement (line 375) is executed. The variables Vector, Reason, and Select_code are returned to the subroutine. Vector and Reason have been explained previously. The third parameter, Select_code are returned to the subroutine. Vector and Reason have been explained previously. The third parameter, Select_code, confirms that the interrupt was a valid one, and identifies the select code of the 98633A Multiprogrammer Interface or 98624A HP-IB Interface on which the interrupt occurred. Before executing the rest of the interrupt handler program code, you must insure that the value of Select_code is a positive number (line 400).

Select_code can be returned as a negative number (line 720), for one of two reasons. The first reason is hardware related: an I/O function enabled to interrupt can receive two external trigger signals (and thus generate two interrupts) too fast for the computer to service them both. So the second interrupt request will be ignored. You must re-enable the interface (line 740) to allow it to interrupt again when the next request occurs. Select_code, in this case, contains a negative select code; for example, if the interrupt came in too fast on select code 29, the value in the variable Select_code would be " – 29".

The second way that a negative value of Select_code can occur is if you have programmed the system incorrectly. If an I/O function is currently enabled to interrupt when its operation is complete (a 14752A Enable_intr statement has been executed), you should program the card only with a Start statement or an External Trigger signal. You should **NOT** execute an Output, Input, Wait_for, or Check_done statement on that Name, since these statements clear the End of Process (EOP) signal on the I/O card that identifies it as the source of interrupt. If the interrupting function cannot be identified, the value of Vector cannot be set, and the negative Select_code will result.

You must also guard against a third case, where the value of Select_code = 0. This indicates that a spurious interrupt occurred: some device pulled the SRQ line, but none of them acknowledged doing it. This condition should never occur during normal operation, so you should abort program execution (lines 795-800)

Multiple Names and Interrupt Handling

When multiple Names are assigned to a 69730A, 69731B, 69770A, or 69771A card and one Name on the card is enabled to interrupt, all the Names on the card are enabled to interrupt. This is because there is only one hardware line per card that can interrupt the computer; it's called Interrupt Request (IRQ). However the Vector that's returned is the Vector of the Name that was in the Enable_intr statement. For example, assume that we have a single 69731B Digital Output card that is assigned two Names: Name1 and Name2. If we enable only Name1 to interrupt and then Start Name2, we will get an interrupt from the card. The Vector that is returned to the interrupt service routine will be the Vector that was assigned to Name1.

There is one more thing to keep in mind when using multiple Names in conjunction with interrupts. If you use any function that does not have to be initialized to "switch" Names (69730A, 69731B, 69751A, 69759A, 69770A, 69771A, 69775A, or 3478A) and more than one Name is enabled to interrupt, then the Vector of the last Name that was Enabled is the Vector that is returned to the interrupt handler routine. For example, Name1 amd Name2 are on the same card and the following sequence is programmed:

```
Enable_intr(Name2,2)
Enable_intr(Name1,1)
Start (Name2)
```

Once an interrupt occurs, the Vector that is returned will be the Vector of Name1; i.e.,"1".

Programming with the Real Time Clock

This section deals with the use of real time clock interrupts, generated by the computer, to pace measurements. It should be noted that applications requiring accurate pacing (such as waveform digitization) should be done with crystal controlled hardware timebase (e.g. 69735A, 69736A, 69750A or Timer function), not the software real time clock.

Pacing a Measurement with the "ON CYCLE" Statement

The Series 200 and 300 BASIC programming lanaguage provides an ON CYCLE statement that causes periodic interrupts. The interrupts can be used to initiate actions (e.g. measurements) in your system applications. The period of the interrupts is specified in the ON CYCLE statement in terms of seconds. More information but the ON CYCLE statement is included in the BASIC Language Reference Manual.

Here is a sample program which takes 10 scanned readings once every minute for one hour:

```
10     COM /Names/ Scan,Adc
20     DIM Whole(1:600)
30     Configure("Menu_on","SCANNER")
40     Init(Adc)
50     ON CYCLE 60 GOSUB Snatch
60     Index=1
70     Minutes=0
80 Loop:IF Minutes<>60 THEN Loop
90       OFF CYCLE
100      FOR I=1 TO 600
110        PRINT Whole(I)
120      NEXT I
130      GOTO Done
140 Snatch:       !This subroutine does the actual data acquisition
150      Input_rblock(Adc,Whole(*),10,Index)
160      Index=Index+10
170      Minutes=Minutes+1
180      RETURN
190 Done:END
```

The 69759A A/D function, named Adc, is used in this example (refer to Chapter 5, Analog Input Functions). It references the Scanner menu named "Scan". The Input_rblock statement (line 150) does all the actual data acquisition. Its parameters are function Name (Adc), data array (Whole(*)), Count(10) and starting Index within the Array. After setting up the system in lines 10 through 70, this program spends most of its time looping on line 80. The "ON CYCLE" statement causes a branch to the Snatch subroutine (lines 140 to 180) once every 60 seconds. The Snatch subroutine packs the ten readings taken from the "Sequential_scan" into a larger array, Whole, which represents the entire hour's worth of data. The Snatch subroutine also increments the Minutes variable, which eventually allows the main program to proceed past line 80. It also increments the Index variable so the hours worth of data can be stored into the array named Whole.

Frontplane Signals and Timing

The 14752A CAT Programming Package simplifies communication between your program and the Multiprogrammer I/O cards. This section deals with the interface of the Multiprogrammer cards with your environment. This interface is described in terms of the card's external edge connector signals; i.e., the frontplane signals. Multiprogrammer I/O cards interface to your system application via the main channel (analog or digital, input or output), control signals, and in some cases, a secondary data bus.

Multiprogrammer cards can also interface with each other the same way.

Main and Secondary I/O Channels

The main channel on any I/O card is, by definition, the primary input or output signals at the card's frontplane edge connector. Information on the main channel of each function is discussed in Chapters 5 through 9 of this manual. Certain I/O cards have a secondary digital data bus for data that is related to a card's operation but is not the main channel. The cards which have secondary data buses are listed below:

Cards with secondary digital inputs	Cards with secondary digital outputs	
69720A	69751A	69775A
69721A	69759A	69790B
69750A	69770A	69791A
69790B	69771A	
69791A		

For a comprehensive discussion of these secondary data buses refer to the applicable I/O card Operating Manual. Any Multiprogrammer card with digital outputs can drive a Multiprogrammer card with digital inputs directly, without computer intervention. Examples of a few of these configurations are:

Card driving	Card receiving	Application
69751A/69759A	69790B/69791A	Buffered A/D
69790B/69791A	69720A/721A	Buffered D/A
69790B/69791A	69750A	Random Scanning

Control Signals

Even the simple I/O card data transfers done by 14752A statements like Output and Input require the cards to perform multiple operations. An explanation of these operations will help in understanding not only how the 14752A statements work, but also how the edge connector signals can be used to interact with the 14752A statements. The next three paragraphs explain the basic operations performed by most output and input cards.

An Output statement performs two basic operations on an output card. First, the data is sent to a holding register on the card called First Rank Storage. Although the card now has the data, it has not yet been sent to the card's edge connector. The second operation consists of "cycling" the card which transfers the data to the card's edge connector via the Second Rank Storage register and output circuits. The cycling operation also generates a gate signal which initiates the gate/flag sequence (handshake cycle). Completion of the gate/flag sequence indicates that the card has completed I/O operations. Whenever an I/O card completes its handshake cycle, the End Of Process (EOP) signal goes high. Since some 14752A statements (Output, Input and Wait_for) must know when a card completes its operations, they use the EOP signal. If the EOP signal is not received within the amount of time specified by the Timeout parameter in the menus, a timeout error will occur.

An Input statement performs two basic operations on an input card. These operations are similar to those just described, but reversed. The first operation consists of "cycling" the card which causes it to take a reading from the edge connector and places it in a holding register called the Input Latch. Although the card now has the data, it has not yet been read into the computer. Cycling an input card also initiates the handshake cycle. Once the card completes its I/O operations, the EOP signal goes high allowing the second operation to start. The second operation is when the data in the Input Latch is sent to the computer.

All I/O cards will automatically complete the handshake cycle when they are done, with the exception of those cards that have a removable gate/flag jumper. With the gate/flag jumper installed (it's shipped that way) the sequence is completed instantaneously. However, removal of this jumper allows an external device to determine the completion of the handskake cycle (refer to the paragraph in this chapter entitled "Gate/Flag Handshake Lines").

All Multiprogrammer cards have a set of control lines which allow communication with the external environment. These control lines allow some of the operations just described to be initiated by an external device. Following is a brief description of the control signals that are common to most of the I/O cards. Some cards may not have these signals and some may have signals not presented here. Refer to the individual card Operating Manuals for edge connector drawings and specifications.

External Trigger (EXT). This input control line allows an external device to cycle the card. For output cards, the leading edge of the External Trigger signal causes data to be transfered from the holding register (First Rank) to the active outputs of the card (Second Rank). For input cards, the leading edge of the External Trigger signal causes the card to take a reading and store it in its Input Latch.

Trigger Polarity Select (TPS). This input determines the logic sense of the EXT input. When the TPS line is driven high (or unconnected), the EXT input is triggered by a low-to-high transition. When the TPS line is driven low (or shorted to ground), the EXT input is triggered by a high-to-low transition.

External Enable (EEN). The External Enable signal controls the active outputs of output cards. When brought low, it causes outputs to be disabled, i.e. all relays open, DAC to zero volts, etc. This is the hardware "safe state" that is in effect after power-up; **not** the INITial values of the menus.

Gate/Flag Handshake Lines. The Gate/Flag handshake insures reliable data transfer between I/O cards and the external application. The Gate output line goes high any time the card is cycled by an External trigger signal or 14752A statement. The 14752A statements that cycle (start) with an I/O card are Output, Input, Start, and in most cases Init. The Gate output signal will go low when Flag

is returned. Although every I/O card has a Gate/Flag handshake, the Gate (GAT) and Flag (FLA) signals are available on the edge connecter of only these cards:

69731B Digital Output
69770A Isolated Digital Input
69771A Digital Input/Analog Comparator

The handshake is provided to insure reliable data transfer when the external device is slow in responding. For example, even though a 69731B Digital Output card can process data instantaneously, the external device connected to the card might not be so fast. In this case, if we send data to the card as fast as the card can take it, we might be sending it to the external device too quickly. To solve this problem, remove the gate/flag jumper to extend the handshake to the external device. The gate signal initiates the data transfer with the external device; the flag signal is returned from the external device when the data transfer has completed and the external device is again ready. The external device can then control when the Digital Output card completes a data transfer, thus, guaranteeing no data will be lost. This is called synchronizing the external device to the Multiprogrammer and computer program. Each data transfer occurs only when both the external device and the computer program are ready for it.

The same kind of thing can happen with the 69770A and 69771A input cards if the external device is slower than the cards. If data is loaded into the Input Latch before the external device can transmit the correct data to the input card, erroneous data will be stored in the Input Latch. Just as with output cards, we can avoid this problem by extending the gate/flag handshake. On input cards, there is an additional input called Flag Edge Select (FES) that allows you to specify whether the data should be stored in the Input Latch on the leading or trailing edge of the Flag signal.

Note that the Gate output is jumpered to the Flag input when you receive the card. The jumper is installed so that the card can operate with external devices which do not use handshaking. The jumper must be left in place if:

a. The external device does not require a Gate signal, or
b. The external device requires a Gate signal, but does not return a Flag signal. When Gate is jumpered to Flag, the Gate pulse width will be approximately 1 microsecond.

Since the gate/flag handshake will not complete until the Flag signal is returned by the external device, a card can remain busy for a long time (or until the Timeout value specified in the menu for that card expires). Therefore, when using the gate/flag handshake, you must be sure that the external device will always respond with a Flag. If for any reason, Flag is not returned, the Output or Input statement associated with the card will not complete, and a timeout error message (code 803) will be generated. Refer to the individual card Operating Manual for the location of the gate/flag jumper.

Flag Polarity Select (FPS). This input determines the logic sense of the FLG input. When the FPS line is driven high (or unconnected), the FLG input is "high true"; i.e. FLG is triggered by a low-to-high transition. When the FPS line is driven low (or shorted to ground), the FLG input is "low true"; i.e. FLG is triggered by a high-to-low transition.

Flag Edge Select (FES). This input line allows you to specify whether the data should be stored in the Input Latch on the leading or trailing edge of the Flag signal. If FES is driven high (or left unconnected), data is latched on the leading edge of Flag. If it is driven low (or shorted to ground), data is latched on the trailing edge of Flag.

Busy (BSY). The Busy signal goes high after the leading edge of Gate and low after the trailing edge of Flag indicating that the function is in process.

End Of Process (EOP). The End Of Process signal indicates the status of the card's operation. If the card has completed its operation, the EOP control line is set high. It can also be tested with the Wait_for and Check_done statements. When these statements detect completion they will set the EOP line low. Also, any 14752A statement that has the ability to "timeout" will check the EOP line and clear it (set EOP low) upon completion. The Get_intr_vector statement also clears EOP. When EOP is used in conjunction with the External Trigger of another card, it can start a second function immediately following the completion of the first, without computer intervention.

Frontplane Signal Timing

Figure 10-5 illustrates the timing relationships of EXT, GAT, FLA, BSY, and EOP. This figure shows the relationships when the card is cycled by an External Trigger (EXT) signal. The sequence is the same when the card is cycled by a 14752A statement (Start, Output, Input) with the exception that the EXT input line does not go high. When a card is cycled, it drives the Gate (GAT) output high indicating that data is available on an output card or data is being requested by an input card. In response to Gate going high, an external device (or on-card jumper) asserts the FLAG (FLA) input (its polarity is determined by FPS) indicating that data has been accepted for an output card. Data is latched by an input card on either the leading or trailing edge of Flag as determined by the Flag Edge Select (FES) signal. When the Flag is asserted, the Gate output goes low. When the Flag goes low, Busy (BSY) goes low and End Of Process (EOP) goes high indicating that the card operation is complete.

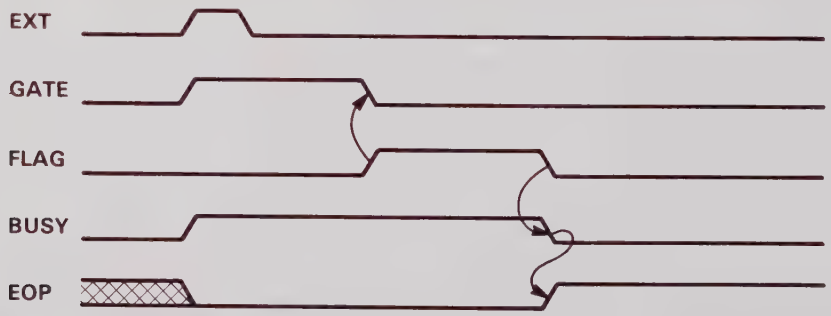


Figure 10-5. Multiprogrammer Frontplane Timing Relationships

More About Multiple Card Functions

The Buffer, A/D, D/A, Scanner, Scan Controller, and Timer menus are " building blocks" for data acquisition systems. These menus allow you to configure multiple card functions in software to suit the needs of your particular system.

The Buffer menu is the basic "building block" for a buffered multiple card function. This menu allows a Memory card or Memory Card System to reference an A/D or D/A card. This is done through a "front-end" linking field in the buffer menu. The newly formed function is referred to as a buffered A/D or buffered D/A function. There is also another function that can be configured in the buffer menu, a multiplexed A/D function. This is a special type of buffered A/D function where a single 69791A Memory Card System references multiple (up to 8) 69759A A/D cards (refer to multiplexed A/D example later in this chapter).

The A/D card menu (69751A/69759A) is the basic "building block" for a scanning multiple card function. This menu allows an A/D card to reference a Scan Control card. This is done through a "Scanner" linking field in the A/D card menu. The newly formed function is referred to as a Scanning A/D function. Note that a buffer can be referencing this A/D card. This function then becomes a buffered scanning A/D multiple card function. For a multiplexed A/D function, each A/D card has the option of referencing a Scan Control card.

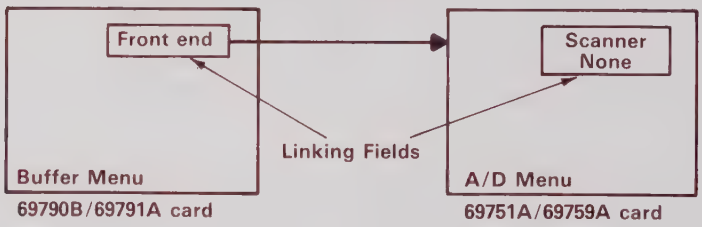
Unlike the Buffer and A/D menus, the Scanner, Scan Controller and D/A menus do not have linking fields that allow front-end card referencing. That is, the Scanner, Scan Controller, and D/A menus can be referenced to form multiple card functions for data acquisition systems, but they can not reference any other cards. Both the Scan Control card and the D/A card are the last cards in the name forwarding structure.

Although both the Scanner and Scan Controller menus can be referenced by an A/D card, the Scanner menu better suits this purpose. This is because the Scanner menu was designed for simple scanning applications. Also, the Scanner menu not only contributes to multiple card functions by being referenced, but it is in itself a multiple card function menu. That is, for random scanning applications the Scanner menu allows you to specify a 69790B Memory card or 69791A Memory Card System for external channel address memory (note that for a buffered scanning A/D function using the 69791A Memory Card System, this can be the same Memory as the buffer). This allows the cards to work together as a "sub system" where the memory can store the random channels to be scanned by the Scan Control card.

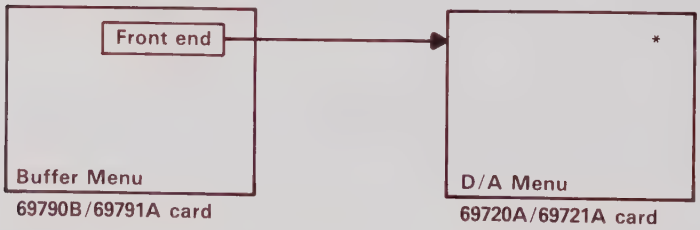
The Timer menu is also a multiple card menu, but unlike other multiple card menus, it can neither reference other cards nor be referenced by other cards. The Timer function is a combination of the 69736A Timer/Pacer, one or two 69775A Counter/Totalizer(s) and an optional 69731B Digital Output card. The entire function is configured in one menu where all the information for every card in the function (such as frame and slot) is specified.

The following figures show how each of the previously described functions are configured. They represent in software how the cards are cabled together in hardware (refer to Appendix C for card interconnections). Note the difference between the menu structures; i.e., the Buffer menu can only reference, the A/D menu can both reference and be referenced, the Scanner, Scan Controller and D/A menus can only be referenced, and the Timer can neither reference nor be referenced by other cards.

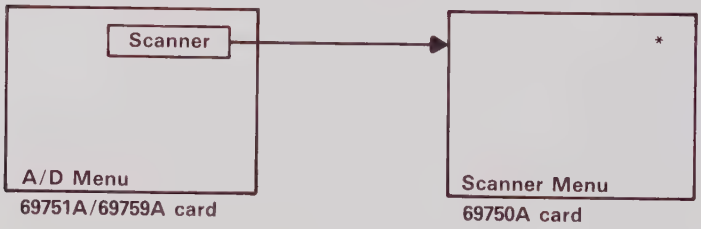
Buffered A/D Function
Function Initialization: Init(Buffer_name)



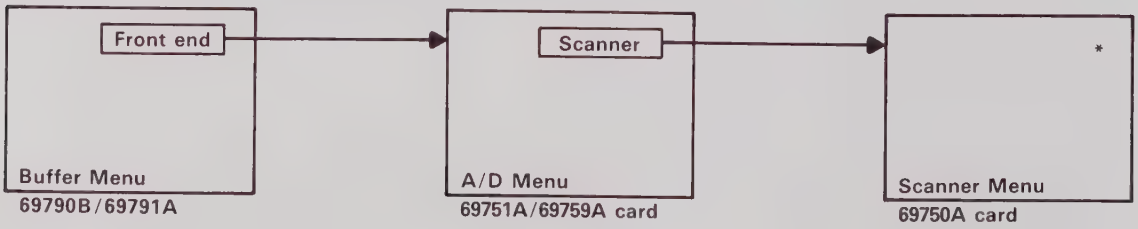
Buffered D/A Function
Function Initialization: Init(Buffer_name)



Scanning A/D Function
Function Initialization: Init(Adc_name)

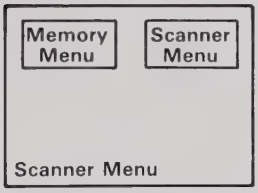


Buffered Scanning A/D Function
Function Initialization: Init(Buffer_name)



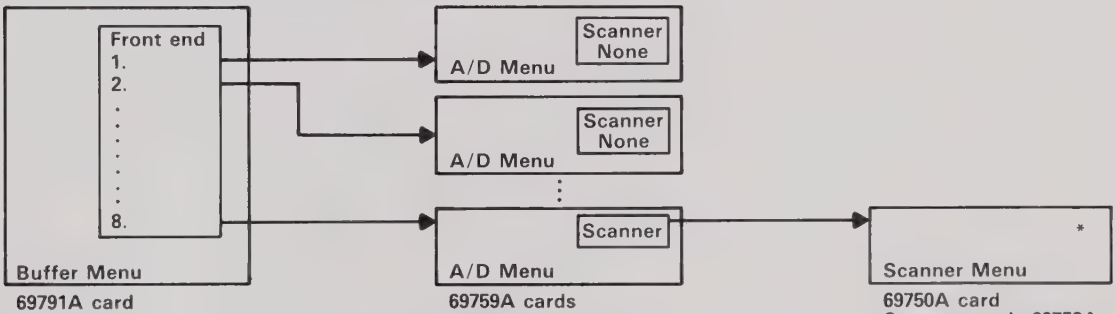
* No Linking Field

The Scanner Function (Random scanning applications only)
Note: When the Scanner is initialized, so is the Memory card.



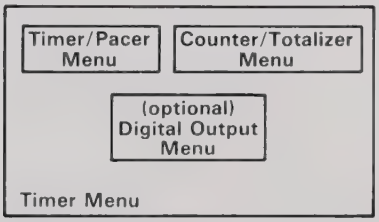
Combines the 69750A Scan Control card with
69790B/69791A Memory card/Memory Card System

Multiplexed A/D Function
Function Initialization: Init(Buffer__name)



69791A card
69759A cards
69750A card
Scanner cards 69752A
69754A
69755A

The Timer Function
Function Initialization: Init(Timer__name)



Combines 69736A Timer/Pacer, one or two 69775A Counter/Totalizer(s)
and (optional) 69731B Digital Output card.

* No Linking Field

Multiplexed A/D Example

Suppose we needed to compute instantaneous input and output power measurements where power in and power out are referenced to two separate isolated commons. A time varying input signal is supplied(refer to system block diagram figure 10-6) . We need a multiplexed A/D function consisting of one 69791A Memory Card System referencing four 69759A 500 kHz A/D cards, and a Timer function consisting of one 69736A Timer/Pacer card, one 69731A Digital Output card, and two 69775A Counter/Totalizer cards. The four A/D cards simultaneously measure the input and output voltage and current, the Memory Card System will store the measurements offline at the precise time intervals specified by the Timer function. Refer to figure 10-7 for the hardware block diagram (Appendix C for card interconnections) and figures 10-9, 10-10 and 10-11 for menu entries.

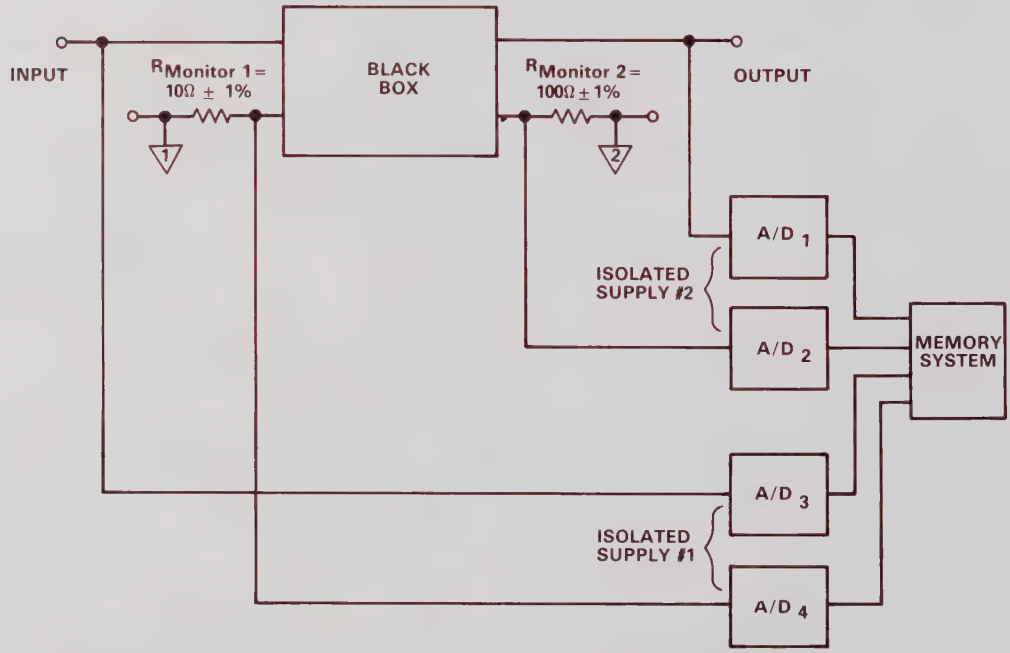


Figure 10-6. System Block Diagram

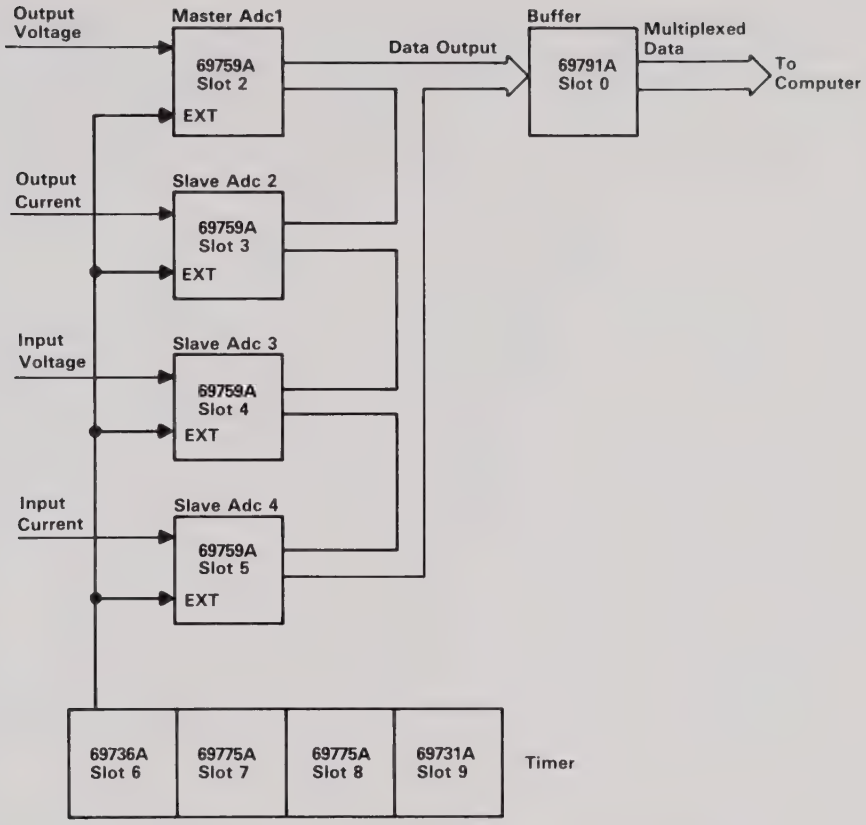


Figure 10-7. Multiplexed A/D Block Diagram (see Appendix C for Card interconnections)

The following figure shows how the menu referencing scheme works for this specific example. Detailed menu configurations for the buffer, Timer, and A/D cards will follow.

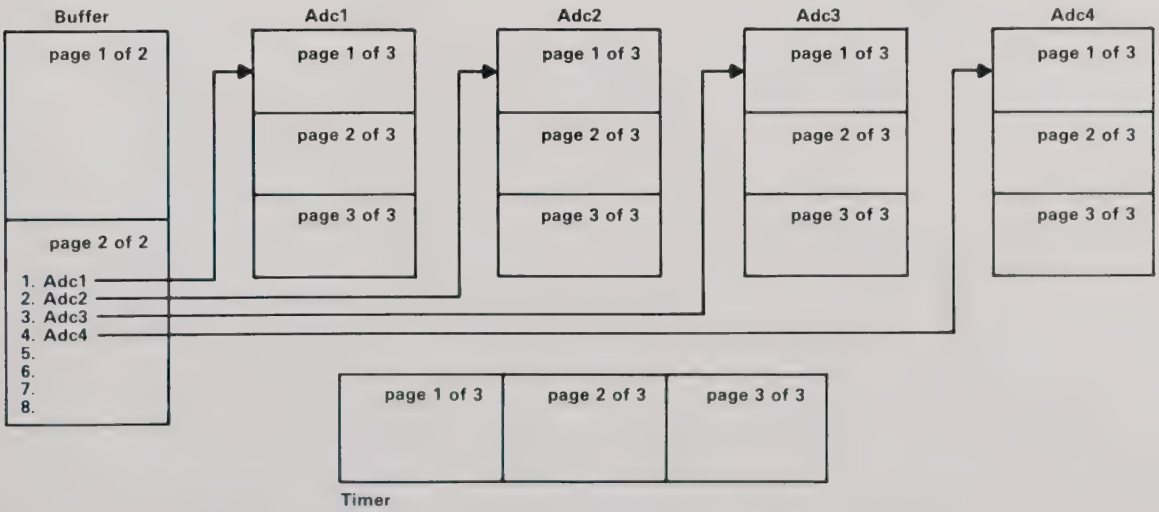


Figure 10-8. Menu Referencing Diagram

Each A/D card's menu specifies the initial modes and data conversion for the card. Only the first A/D card menu is shown here. The other three A/D card menus are identical except for the function Name and slot. Note that the Multiplex Control Circuitry of each A/D card **MUST** be enabled in order to allow each A/D to multiplex data.

```
Name --- Adc1                        Date: 21 Aug 1984
Model 69759A 500 KHz A/D             Time: 16:56:38
File Name -- Mux_buf_ad
User ID ----
Time-stamp - First Time

                                  Page 1 of 3
Multiprogrammer type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot .......... 2
Full-scale Range  ........ +-10.24 volts
Initial Internal Range ... 10

Range Source ............. Internal
Scanner .................. None
Timeout .................. 5
```

```
Name --- Adc1                        Date: 21 Aug 1984
Model 69759A 500 KHz A/D             Time: 16:57:12
File Name -- Mux_buf_ad
User ID ----
Time-stamp - First Time

                                  Page 2 of 3

Data Conversion ........... Standard
```

Figure 10-9. Menu Entries for the 69759A A/D Card

```
Name --- Adc1                          Date: 21 Aug 1984
Model 69759A 500 KHz A/D               Time: 16:57:43
File Name -- Mux_buf_ad
User ID ----
Time-stamp - First Time

                                    Page 3 of 3

Internal Trigger ..... Disabled
External Trigger ..... Enabled
Trigger Mode ......... Multiple
Trigger Polarity ..... Negative
Lockout Polarity ..... Disabled
Master Output Enable ..... Enabled
Mux Output Control ....... Enabled
External Output Enable .... Disabled
Gate Mode................. Multiple
Return Data With ..... Sign Extension
```

Figure 10-9. Menu Entries for the 69759A A/D Card (continued)

The buffer menu references each A/D card. This allows the cards to be initialized as a subsystem and the data to be post processed. That is, the data is converted according to the data conversion specified in each A/D card's menu. The buffer menu has no associated data conversion . It is simply used as a storage device for the A/D cards. Note that the first A/D name selected is the Master of the multiplexed chain, the second is the closest Slave to the Master, and so on down the chain.

```
Name --- Buffer                    Date: 21 Aug 1984
Model Buffer                       Time: 16:59:27
File Name -- Mux_buf_ad
User ID ----
Time-stamp - First Time

                                   Page 1 of 2

Multiprogrammer type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot .......... 0

Initial Mode .......... FIFO
Initial Lockout ....... Off

Timeout ....... 5
```

```
Name --- Buffer                    Date: 21 Aug 1984
Model Buffer                       Time: 17:02:18
File Name -- Mux_buf_ad
User ID ----
Time-stamp - First Time

                                   Page 2 of 2
Memory Type .... 69791A  No of Extenders ... 0
Ref Reg 1 .. 40000       Ref Reg 2 ... 0
Buffer Direction .. In   Front End Type .. 69759A
A/D Names:    1 . Adc1
              2 . Adc2
              3 . Adc3
              4 . Adc4
              5 .
              6 .
              7 .
              8 .
```

Figure 10-10. Menu Entries for the Buffer function.

The Timer menu configures the 69736A Timer/Pacer, the 69775A Counter/Totalizers, and the 69731A Digital Output card as a card set. This forms the Timer function.

```
 Name --- Timer                    Date: 21 Aug 1984
 Model Timer                       Time: 16:53:59
 File Name -- Mux_buf_ad
 User ID ----
 Time-stamp - First Time

                                 Page 1 of 3

Multiprogrammer type ... HP6944A
Interface Select Code .. 29

Timer/Pacer Frame .. 0
Timer/Pacer Slot ... 6
Counter Frame ..... 0
Counter Slot ..... 7

Counter size ... 31 Bits
  Extension Counter Frame ... 0
  Extension Counter Slot .... 8
Timeout ........ 5
```

```
 Name --- Timer                    Date: 21 Aug 1984
 Model Timer                       Time: 16:54:23
 File Name -- Mux_buf_ad
 User ID ----
 Time-stamp - First Time

                                 Page 2 of 3


Data Conversion ........... Standard


Run Time Limit Checking ... No
```

Figure 10-11. Menu Entires for the Timer Function

```
Name --- Timer                          Date: 21 Aug 1984
Model Timer                             Time: 16:55:17
File Name -- Mux_buf_ad
User ID ----
Time-stamp - First Time

                                  Page 3 of 3

Initial Period ........ .001
Initial Count ......... 40000

Digital Out Controller .. Yes
     Controller Frame  ............ 0
     Controller Slot .............. 9

     Timer Mode ................... Burst
     Reference Trigger Polarity.... Positive
```

Figure 10-11. Menu Entries for the Timer Function (continued)

Multiplexed A/D Example Program Listing

In the following example, the outputs of four A/D cards are multiplexed into one Memory Card System. This data is then demultiplexed using the data conversion of the specified A/D card and stored into individual arrays by the Demultiplex statement.

```
10    OPTION BASE 1
20    COM /Names/ Timer,Adc1,Adc2,Adc3,Adc4,Buffer
30    Configure("MENU ON","Muxd_adc")
40    DIM Results1(10000),Results2(10000),Results3(10000),Results4(10000)
50    DIM I_in(10000),I_out(10000),P_in(10000),P_out(10000)
60    Rmonitor1=10
70    Rmonitor2=100
80    !
90    ! Results is a two dimensional array where 10000 is the number of
100   ! readings for each A/D card and 4 is the number of A/D cards in the
110   ! system. Each dimension can be any number less than 32768 but the
120   ! product must equal the desired total number of readings.
130   !
140   INTEGER Results(10000,4)
150   !
160   ! Initialize all cards in the Timer function and insure that the Timer
170   ! is not outputting pulses.
180   !
190   Init(Timer)
200   !
210   ! Initialize the Buffer function as a subsystem. This includes
220   ! initialization of the four A/D cards as well as the Memory Card System.
230   !
240   Init(Buffer)
250   Start(Timer)
260   !
270   ! Wait for the value specified for reference register 1 to be reached.
280   !
290   Wait_for(Buffer)
300   !
310   ! Read a block of multiplexed data from the Memory Card System into
320   ! the Integer array named Results. Each A/D card will supply 10000
330   ! readings to give a total block size of 40000.
340   !
350   Input_iblock(Buffer,Results(*))
360   !
370   ! Demultiplex the data of a specified A/D card from the array named
380   ! Results by converting the data according to the specified A/D card's
390   ! data conversion, and returning the processed data into the associated
400   ! ResultsX array.
410   !
420   Demultiplex(Buffer,Adc1,Results(*),Results1(*))
430   Demultiplex(Buffer,Adc2,Results(*),Results2(*))
440   Demultiplex(Buffer,Adc3,Results(*),Results3(*))
450   Demultiplex(Buffer,Adc4,Results(*),Results4(*))
460   !
470   ! Calculate Power in and Power out by measuring Voltage in,
480   ! Current in, Voltage out and Current out.
490   !
500   MAT I_in= Results4/(Rmonitor1)
510   MAT P_in= Results3 . I_in
520   MAT I_out= Results2/(Rmonitor2)
530   MAT P_out= Results1 . I_out
540   !
550   ! Print Power in and Power out results.
560   !
570   PRINT P_in(*)
580   PRINT P_out(*)
590   END
```

Buffered Random Scanning A/D with External Ranges Example

Suppose we want to test electronic equipment at specific channel addresses where each channel measurement requires a different range. We can do so by randomly scanning specified channels and externally supplying the range values for each. By supplying random channel addresses to be scanned, we can test a variety of devices without altering the system configuration. This is illustrated by the following example. Refer to Figure 10-12 for a hardware block diagram (Appendix C for card interconnections) and Figure 10-13, 10-14, 10-15, and 10-16 for menu entries.

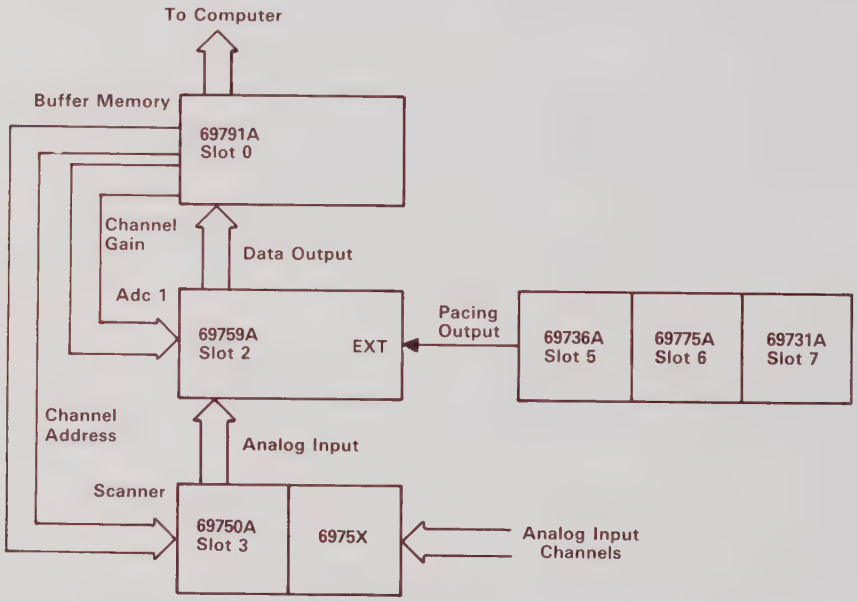


Figure 10-12. Buffered Random Scanning A/D block Diagram (see Appendix C for card interconnections)

```
Name --- Scanner                    Date: 21 Aug 1984
Model Scanner                       Time: 17:08:55
File Name -- Buf_scn_ad
User ID ----
Time-stamp - First Time

                                  Page 1 of 2

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot .......... 3
Scanner Mode ..... Random
    Memory Frame ....... 0
    Memory Slot ........ 0
    Memory Type ........ 69791A
Timeout ....... 5
```

```
Name --- Scanner                    Date: 21 Aug 1984
Model Scanner                       Time: 17:09:22
File Name -- Buf_scn_ad
User ID ----
Time-stamp - First Time

                                  Page 2 of 2


Initial Switch State ..... Enabled

Stop Channel Interrupt ............... Disabled
External Trigger Lockout Interrupt .. Disabled

Pacing Source ............. External
```

Figure 10-13. Menu Entries for the Scanner Function

```
 Name --- Adc                        Date: 21 Aug 1984
 Model 69759A 500 KHz A/D            Time: 17:10:00
 File Name -- Buf_scn_ad
 User ID ----
 Time-stamp - First Time

                                  Page 1 of 3
Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot .......... 2
Full-scale Range  ........ +-10.24 volts
Initial Internal Range ... 10

Range Source ............. External
Scanner .................. Scanner
Timeout .................. 5
```

```
 Name --- Adc                        Date: 21 Aug 1984
 Model 69759A 500 KHz A/D            Time: 17:10:51
 File Name -- Buf_scn_ad
 User ID ----
 Time-stamp - First Time

                                  Page 2 of 3


Data Conversion ........... Standard
```

Figure 10-14. Menu Entries for the 69759A A/D Card (continued)

```
Name --- Adc                          Date: 21 Aug 1984
Model 69759A 500 KHz A/D              Time: 17:11:51
File Name -- Buf_scn_ad
User ID ----
Time-stamp - First Time

                                   Page 3 of 3

Internal Trigger ..... Disabled
External Trigger ..... Enabled
Trigger Mode ......... Multiple
Trigger Polarity ..... Negative
Lockout Polarity ..... Disabled
Master Output Enable ..... Enabled
Mux Output Control ....... Disabled
External Output Enable .... Disabled
Gate Mode................. Multiple
Return Data With ..... Sign Extension
```

Figure 10-14. Menu Entries for A/D Card (continued)

```
Name --- Buffer                          Date: 21 Aug 1984
Model Buffer                             Time: 17:12:54
File Name -- Buf_scn_ad
User ID ----
Time-stamp - First Time

                                    Page 1 of 2

Multiprogrammer type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot .......... 0

Initial Mode ........... FIFO
Initial Lockout ....... Off

Timeout ....... 5
```

```
Name --- Buffer                          Date: 21 Aug 1984
Model Buffer                             Time: 17:13:49
File Name -- Buf_scn_ad
User ID ----
Time-stamp - First Time

                                    Page 2 of 2
Memory Type .... 69791A   No of Extenders ... 0
Ref Reg 1 .. 0            Ref Reg 2 ... 65535
Buffer Direction .. In    Front End Type .. 69759A
A/D Names:    1 . Adc
              2 .
              3 .
              4 .
              5 .
              6 .
              7 .
              8 .
```

Figure 10-15. Menu Entries for the Buffer Function

```
Name --- Timer                    Date: 21 Aug 1984
Model Timer                       Time: 17:06:50
File Name -- Buf_scn_ad
User ID ----
Time-stamp - First Time

                                  Page 1 of 3
Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Timer/Pacer Frame .. 0
Timer/Pacer Slot ... 5
Counter Frame ..... 0
Counter Slot ..... 6

Counter size ... 15 Bits


Timeout ........ 5
```

```
Name --- Timer                    Date: 21 Aug 1984
Model Timer                       Time: 17:07:03
File Name -- Buf_scn_ad
User ID ----
Time-stamp - First Time

                                  Page 2 of 3


Data Conversion ........... Standard


Run Time Limit Checking ... No
```

Figure 10-16. Menu Entries for the Timer Function

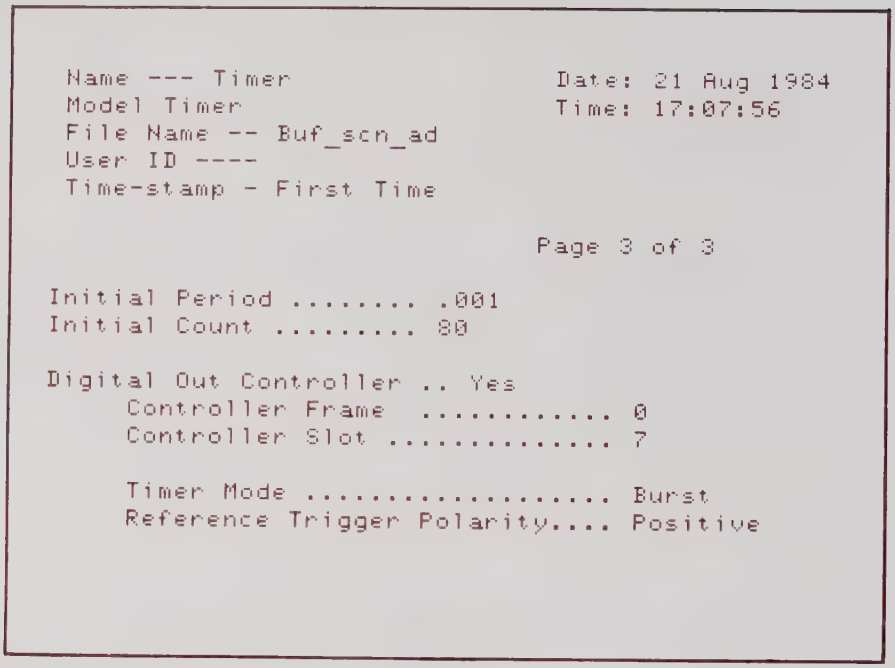


Figure 10-16. Menu Entries for the Timer Function (continued)

Buffered Random Scanning with External Ranges Program Listing

The following program is used to scan eight random channels and store the measurements in a 69791A Memory Card System. Each measurement has an associated range value which is supplied externally. The Load_channels statement is used to download the random channel addresses and associated range values to the memory buffer (note that the same Memory Card System is used for both the external channel address memory and the data buffer). The Input_iblock statement is used to retrieve integer data from the buffer and the Translate statement is used to process this data using the supplied ranges and the data conversion of A/D card.

```
10    COM /Names/ Timer,Scanner,Adc,Buffer
20    Configure("MENU ON","Buf_scn_ad")
30    !
40    ! Define the channels for the random scan
50    !
60    DATA 5,3,1,7,6,2,0,4
70    !
80    ! Define the corresponding ranges for the
90    ! 69759A A/D card.
100   !
110   DATA .1,1,10,10,10,.1,.1,1
120   !
130   ! Dimension the required arrays.
140   !
150   DIM Channels(8),Ranges(8),Results(80)
160   INTEGER Temp(80)
170   !
180   ! Fill the channel array.
190   !
200   READ Channels(*)
210   !
220   ! Fill the range array.
230   !
240   READ Ranges(*)
250   !
260   ! Initialize the cards in the Timer function
270   ! to the configuration specified in the
280   ! menu and turn off its output.
290   !
300   Init(Timer)
310   !
320   ! Initializing the Buffer causes the A/D and
330   ! Scanner to be initialized due to the
340   ! forward references in the menus.
350   !
360   Init(Buffer)
370   !
380   ! The Load_channels statement merges the
390   ! channel and range arrays, stores them at
400   ! the top of memory in the Memory card
410   ! system, sets the read pointer and reference
420   ! register 2 to the first channel, sets
430   ! reference register 1 to the last channel,
440   ! and sets the memory mode to simultaneous
450   ! recirculate.
460   !
470   Load_channels(Scanner,Channels(*),Ranges(*))
480   !
490   ! The Start statement is used to start the
500   ! Timer function. An external trigger to the
```

Example Programming Listing (continued)

```
510   ! 69736A card in the Timer function could
520   ! have been used instead.
530   !
540   Start(Timer)
550   !
560   ! Wait for the Timer to output its 80 pulses.
570   !
580   Wait_for(Timer)
590   !
600   ! Prepare the Memory card system for a block
610   ! input.
620   !
630   Set_recirc_in(Buffer)
640   Write_rpointer(Buffer,0)
650   !
660   ! Get the raw data into an INTEGER array.
670   !
680   Input_iblock(Buffer,Temp(*))
690   !
700   ! Get the data into a REAL array using the
710   ! data conversion of the A/D card and the
720   ! ranges supplied during the measurement.
730   !
740   Translate(Adc,Temp(*),Results(*),Ranges(*))
750   !
760   ! Display the results.
770   !
780   PRINT Results(*)
790   END
```

Continuous Data Acquisition

Large amounts (blocks) of data can be input into the computer using the 69790B Memory card or 69791A Memory Card System, or the Buffer menu described in Chapter 9. If the data is read from memory fast enough, more data can be written back in at a continuous rate by an external device. This is called Continuous Data Acquisition. In order to implement Continuous Data Acquisition, the rate at which data is transferred to the computer (read from buffer memory) MUST be greater than the rate at which data is being collected (written to buffer memory). The faster the transfer rate, the more "free time" the computer has to do other things (like storing the data on a disc or processing the data) before it must once again empty the buffer memory.

In this section we will develop some formulas which will help you choose values for particular transfer parameters to maximize the computer's free time. The transfer rate and maximum size of the buffer are fixed according to the memory type. Refer to the following chart for the buffer size and transfer rate values:

Memory Type	Buffer Size	Transfer Rate with DMA	Transfer Rate without DMA
69790B	4095 words	—	15 k words/sec
69791A (no 69792A)	65535	200 k word/sec	30 k
69791A + (1) 69792A	262143	180 k	30 k
69791A + (2) 69792A	458751	180 k	30 k
69791A + (3) 69792A	655359	180 k	30 k
69791A + (4) 69792A	851967	180 k	30 k
69791A + (5) 69792A	1048575	180 k	30 k

The transfer parameters are described as follows:

Acquistion Rate — The rate at which new data is written into buffer memroy by an external device. For most cases, this is the rate that the function is paced by a 69735A, 69736A, or Timer function.

Reference Count — The number of data words that a Memory card/Memory Card System will collect before telling the program that is almost full. This is reference register 1 (ref reg 1) for functions using the 69791A.

Free Time — The amount of time it takes the Memory card/Memory Card System to acquire enough data to reach the reference count. During this time the computer is free to do other tasks.

Latency Time — The time from when the reference count is reached to when the computer starts reading data from the buffer memory.

Transfer Block Size — The size of the array that the computer will fill each time it reads the buffer memory.

Transfer Time — The time that it takes the computer to read all the data in the transfer block.

NOTE

Even though the Buffered A/D functions can be paced at maximum acquisition rates of up to 500 kHz, these rates are much higher than the transfer rate. Therefore, these functions should only be paced at rates lower than the transfer rate for continuous data acquisition.

The cycle that occurs during continuous data acquisition can be broken down into three time periods; free time, latency time, and transfer time (Refer to Figure 10-17). At the end of the free time period, the buffer memory indicates to the computer that it has reached the reference count. The program. then begins to take action. At the end of the latency time, the program begins reading data from the buffer memory, until the buffer memory is emptied by the end of the transfer time.

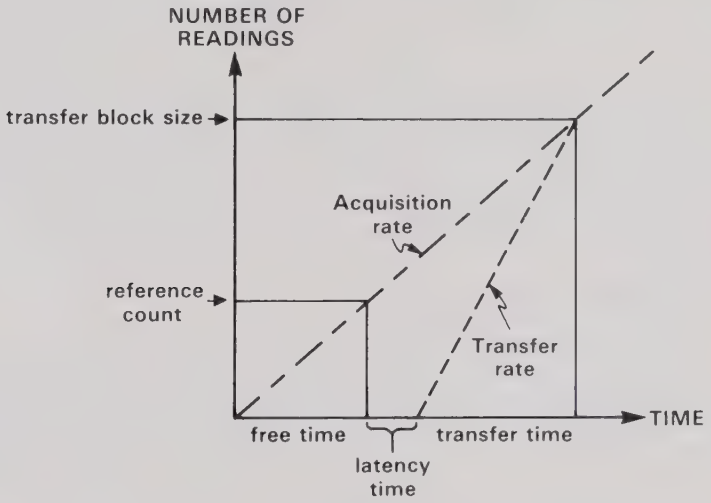


Figure 10-17. Continuous Data Acquisition

Calculating the Transfer Block Size

The number of readings that can be stored in the Memory card/Memory Card System places a limition on the transfer block size. This limitation is also a function of the aquisition and transfer rates:

Maximum transfer block size = Buffer size/1-(acquisition rate/transfer rate)

Use of the maximum transfer block size minimizes the total number of blocks required for a given number of readings. This has the effect of reducing the per block overhead (latency time and program looping statements) and produces longer periods of free time. Smaller transfer block sizes can be used as long as the resultant period of free time is sufficient to process the block.

Latency Time

At some point the Memory card or Memory Card System will indicate to the program that it has reached a level specified by the reference count. The time required for the program to respond to this indication and actually start taking data from memory is a function of the program statements used. The interrupt structure can be used to service the Memory card/Memory Card System, however, simple I/O statements (Wait_for and Check_done) will produce latency times which are an order of

magnitude less than those encountered with the interrupt system. In our example, the Wait_for and Input_iblock statements minimize the latency time:

```
.
.
.
Wait_for (Buffer)
Input_iblock (Buffer,Results(*)
.
.
.
```

Here, we must allow 3.9 ms for the Wait_for statement to complete, after detecting function complete, and 2.5 ms entry overhead for the Input_iblock. The total latency time is therefore 6.4 ms. If interrupts were being serviced in the background of this program, the worst case execution time of the interrupt service routines would be part of the latency time.

Calculating the Reference Count

The 69790B Memory card, 69791A Memory Card System or Buffer should be configured as a FIFO input buffer. In this mode, the Memory card/Memory Card System will indicate function completion when the differnce between the write pointer and the read pointer is equal to the reference count specified. The reference count is calculated so that the block transfer completes just as the last reading arrives:

Reference Count = Transfer block size *(1 - (acquisition rate)/(transfer rate)) - (latency time) * (acquisition rate)

Note that the above equation is only used if the transfer block size is less than the maximum block size. For the maximum transfer block size, the above equation simplifies to:

Reference Count = buffer size - (latency time) * (aquisition rate)

Calculating the Free Time

The free time can then be calculated by using the following equation:

Free Time = (transfer block size)/(acquisition rate) - (transfer block size)/(transfer rate) - (latency time)

Example of Continuous Data Acquisition

Suppose we wanted to measure the roundness of a sample which is rotating at 1 rev/sec. A total of 2000 samples will be taken from a transducer for each revolution. If the output of the transducer is outside the range of 0.0600 volts to 0.0850 volts, the sample must be rejected. This is accomplished by pulsing a solenoid with the Relay card. For this example, the transfer rate is fixed (200,000 readings per second) and the maximum size of the buffer is fixed (65535 readings for the 69791A Memory card with no extenders)*. Note that the transfer rate would decrease to 30,000 readings/sec without a DMA card. The maximum transfer block size, reference count and free time can be calculated as follows:

$$\text{Maximum transfer block size} = \frac{65535}{1-(2000 \text{ samples per second})/(200{,}000 \text{ samples per second})} = 66,197$$

Latency time = 6.4 ms

Reference Count = 2000 • (1-2000 samples per second /200,000 samples per second) — (.0064 seconds • 2000 samples per second)

= 1967

This is the value (rounded down) that we must enter in the menu for Ref Reg1.

Free Time = (2000 samples)/(2000 samples per second) — (2000 samples)/200,000 samples per second) — .0064

= .9836 sec

This is more than the amount of time required to process one block of readings (.15 sec) in lines 610 through 760. Note that if we did not have enough free time , we would have to go back and use a lower value for the acquisition rate or a larger block size.

* This example requires a 98620B DMA Controller card in the Series 200 computer.

```
Name --- Adc                          Date: 21 Aug 1984
Model 69759A 500 KHz A/D              Time: 17:18:59
File Name -- Cnt_buf_ad
User ID ----
Time-stamp - First Time

                                  Page 1 of 3

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot .......... 2
Full-scale Range  ........ +-10.24 volts
Initial Internal Range ... 1

Range Source ............. Internal
Scanner .................. None
Timeout .................. 5
```

```
Name --- Adc                          Date: 21 Aug 1984
Model 69759A 500 KHz A/D              Time: 17:19:36
File Name -- Cnt_buf_ad
User ID ----
Time-stamp - First Time

                                  Page 2 of 3


Data Conversion ........... Standard
```

Figure 10-18. Menu Entries for the 69759A A/D Card

```
Name --- Adc                          Date: 21 Aug 1984
Model 69759A 500 KHz A/D              Time: 17:20:01
File Name -- Cnt_buf_ad
User ID ----
Time-stamp - First Time

                                   Page 3 of 3

Internal Trigger ..... Disabled
External Trigger ..... Enabled
Trigger Mode ......... Multiple
Trigger Polarity ..... Negative
Lockout Polarity ..... Disabled
Master Output Enable ..... Enabled
Mux Output Control ....... Disabled
External Output Enable .... Disabled
Gate Mode................. Multiple
Return Data With ..... Sign Extension
```

Figure 10-18. Menu Entries for the 69759A A/D Card (continued)

```
Name --- Buffer                          Date: 21 Aug 1984
Model Buffer                             Time: 17:21:56
File Name -- Cnt_buf_ad
User ID ----
Time-stamp - First Time

                                  Page 1 of 2


Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot .......... 0

Initial Mode .......... FIFO
Initial Lockout ....... Off

Timeout ....... 5
```

```
Name --- Buffer                          Date: 21 Aug 1984
Model Buffer                             Time: 17:22:32
File Name -- Cnt_buf_ad
User ID ----
Time-stamp - First Time

                                  Page 2 of 2
Memory Type .... 69791A  No of Extenders ... 0
Ref Reg 1 .. 1967        Ref Reg 2 ... 0
Buffer Direction .. In   Front End Type .. 69759A
A/D Names:    1 . Adc
              2 .
              3 .
              4 .
              5 .
              6 .
              7 .
              8 .
```

Figure 10-19. Menu Entries for the Buffer Function

```
Name --- Relay                      Date: 21 Aug 1984
Model 69730A Relay Actuator        Time: 17:23:21
File Name -- Cnt_buf_ad
User ID ----
Time-stamp - First Time

                                   Page 1 of 2

Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot .......... 4
Start Bit ..... 0
Length ........ 1

Timeout ....... 5
```

```
Name --- Relay                      Date: 21 Aug 1984
Model 69730A Relay Actuator        Time: 17:24:53
File Name -- Cnt_buf_ad
User ID ----
Time-stamp - First Time

                                   Page 2 of 2

Standard Data Type ........ Unsigned Binary
Data Conversion ........... Standard

Run Time Limit Checking ... No

Initial Value ............. 0
```

Figure 10-20. Menu Entries for the 69730A Relay Card

```
Name --- Pacer                          Date: 21 Aug 1984
Model 69736A Timer/Pacer                Time: 17:17:20
File Name -- Cnt_buf_ad
User ID ----
Time-stamp - First Time

                                      Page 1 of 2


Multiprogrammer Type ... HP6944A
Interface Select Code .. 29

Frame ......... 0
Slot .......... 3



Timeout ....... 5
```

```
Name --- Pacer                          Date: 21 Aug 1984
Model 69736A Timer/Pacer                Time: 17:17:59
File Name -- Cnt_buf_ad
User ID ----
Time-stamp - First Time

                                      Page 2 of 2

Data Conversion ........... Standard


Run Time Limit Checking ... No


Initial Value ............. 5.E-5
Initial Mode .............. Continuous
```

Figure 10-21. Menu Entries for the 69736A Timer/Pacer Card

```
10     COM /Names/ Pacer,Adc,Buffer,Relay
20     Configure("MENU ON","Cnt_buf_ad")
30     !
40     ! Set up INTEGER limits for quick
50     ! comparisons of raw data.
60     !
70     INTEGER Low_limit,High_limit
80     !
90     ! If the A/D is set to the +/-1 volt
100    ! range, then each count of the raw data
110    ! corresponds to 0.0005 volts. Therefore,
120    ! Low_limit=120 corresponds to 0.0600 volts
130    ! and High_limit=170 corresponds to 0.0850
140    ! volts.
150    !
160    Low_limit=120
170    High_limit=170
180    !
190    ! Dimension the required array.
200    !
210    Block_size=2000
220    INTEGER Results(1:2000)
230    !
240    ! Initialize the Pacer card to the
250    ! configuration specified in the menu
260    ! and turn off its output.
270    !
280    Init(Pacer)
290    !
300    ! Initializing the Buffer causes the A/D
310    ! to be initialized due to the forward
320    ! reference in the menus.
330    !
340    Init(Buffer)
350    !
360    ! Initializing the Relay card turns the
370    ! output off.
380    !
390    Init(Relay)
400    !
410    ! Flag is used to record any out of spec
420    ! data points. It is initialized to 0.
430    !
440    Flag=0
450    !
460    ! The Start statement is used to start the
470    ! Pacer. An external trigger to the
480    ! 69736A card could have been used instead.
490    !
500    Start(Pacer)
510  Block_loop: !
520    !
530    ! Wait for the Buffer to reach the level
540    ! specified by reference register 1.
550    !
560    Wait_for(Buffer)
570    !
580    ! Get the raw data into an INTEGER array.
590    Input_iblock(Buffer,Results(*))
600    !
610    ! Test the data for out of spec and set Flag
620    ! if an error is found.
```

Figure 10-22. Continuous Data Acquisition Example Program Listing

```
630   !
640   IF MIN(Results(*))<Low_limit THEN Flag=1
650   IF MAX(Results(*))>High_limit THEN Flag=1
660   !
670   ! Use Flag to create a 0.1 second pulse on the Relay
680   ! card to indicate out of spec condition.
690   !
700   Output(Relay,Flag)
710   WAIT .1
720   Flag=0
730   Output(Relay,Flag)
740   ! Continue to acquire data.
750   !
760   GOTO Block_loop
770   END
```

Figure 10-22. Continuous Data Acquisition Example Program Listing (continued)

In this example, the free time (lines 600-750) is spent testing the lower and upper limits of the Results array for the values out of spec.

Writing to Disc

Free time could be spent writing data to the disc. This would require you to add a few new statements to your application program.

First, create a BDAT file in which to store the data **before** any data acquisition is done:

```
CREATE BDAT "DATA", Filesize
  ASSIGN @ Disc to "DATA"
```

File size (# of 256 byte records) (rounded up) = (# of bytes/reading) * (# of readings/xfer) * (# of transfers) / (#of bytes/record)

where (# of bytes/reading) = 2 for Input_iblock and 8 for Input_rblock.

Next, write data to the disc **after** the data is acquired (inside the data acquisition loop):

```
OUTPUT @ Disc; Results (*)
```

Finally, close the disc file **after** the data acquisition loop:

```
ASSIGN @ Disc to *
```

Register Reference Using Peek and Poke

The 14752A CAT Programming Package removes from the user the burden of dealing with Multiprogrammer cards on a select code, slot address, and card subaddress basis. The Package supports every capability of each card, so that dealing with the cards in this manner is unnecessary. However, the statements Peek and Poke have been included to allow access to the lowest level programming of the card.

NOTE

The 14752A statements make certain assumptions about the known state of a function based on the way it was programmed via other statements. No guarantee can be made that the statements will work after a function has been programmed using Peek or Poke.

The Peek statement will read back data from an I/O card subaddress into the variable "Value". The Poke statement will write the Value to an I/O card subaddress. The syntax for these commands is as follows:

Peek(Name,Subaddress,Value)
Poke(Name,Subaddress,Value)

Name is the standard 14752A function Name as used throughout this manual. Name must represent a single Multiprogrammer I/O card; multiple card functions and the 3478A cannot be Peeked or Poked.

Subaddress is the specific subaddress on the individual card (0 through 3). The registers on card 2 of the 69791A Memory card are accessed by using subaddresses 4 through 7. What Subaddress means is card specific. Refer to the I/O card Operating Manuals for the register subaddresses which can be programmed or read back. If an attempt is made to write to a subaddress that is not on a card, no error is generated and the statement is ignored. If a subaddress not on the card is read, no error is generated and the data value returned is meaningless.

Value is a 16-bit two's complement REAL number. Values Peeked or Poked will not use Data Conversion or Run Time Limit Checking. Bits which are not connected (do not exist) on the card are generally "Peeked" as "1"s. Peek and Poke operate only on the entire register, regardless of Start Bit or Length.

Chapter 11
14752A Statement Summary

This chapter contains two different lists of all the 14752A statements. The first list is arranged by function. You can use this list to determine which 14752A statements are supported for a particular I/O function. The second list is arranged alphabetically and gives a brief description of each statement. The numbers in the boxes next to the statements in the second listing refer to the Chapter(s) where more detailed information can be found about each statement. Note: all statements are keyboard executable after the "Configure" statement has been executed. To execute any of these statements from the keyboard, you must preface the statement with "CALL" ; e.g. CALL Output(Dac,10). The terms and conditions used in both of these listings are as follows:

[ ] - items enclosed in brackets are optional.

, - all parameters are separated by commas.

Filename - A file name consists of one to ten characters. Series 200 and 300 BASIC file names can contain uppercase letters, lowercase letters, the underbar(_), and CHR$(161) thru CHR$(254).

Variable Names - A name consists of one to fifteen characters. The first character must be an uppercase ASCII letter or one of the characters from CHR$(161) thru CHR$(254). Remaining characters, if any, can be upper or lowercase ASCII letters, numerals, underbar (_), or CHR$(161) thru CHR$(254). Names may be typed using any combination of uppercase and lowercase letters, unless the names use the same letters as a BASIC keyword. Keywords are recognized whether they are typed in uppercase or lowercase. Therefore, to use the letters of a keyword as a name, the name entered must contain some *mixture* of uppercase and lowercase letters when typing in the name.

Function Names - These are special variable names that are referred to as "Name" in this manual. Name is used by each 14752A statement to identify the I/O functions upon which the specified action will be performed. All of these Names **MUST** be included in the "COM/Names/..." statement at the beginning of your application program. The 14752A Configure statement assigns its own numeric value to Name variables, therefore you must never use a function Name in any BASIC statement that will assign a value to it.

CAUTION

Never use a configured Name in a BASIC statement that will assign a value to it. For example, never use a BASIC statement like "Dvm = 10" when Dvm is a function name.

Numeric Variable Names - Can be any valid variable name as described above. In BASIC, numeric variables are either of two types, INTEGER or REAL.

The numeric variables used in the two lists of statements in this chapter are all of type REAL. For limitations on the range of allowable values that can be assigned to these variables, consult the appropriate chapter. The following is a list of the numeric variable names that are used:

Value	Vector	Count	Channel	Voltage_value
Index	Ref_count	Start_chan	Stop_chan	Current_value
Period	Range	Repeat	Flag	

A REAL is a numeric data type that has a range of values from approximately: $-1.797693134862315E+308$ thru $-2.225073858507202E-308$ and $+2.225073858507202E-308$ thru $+1.797693134862315E+308$. If a numeric variable is not explicity declared as an INTEGER, it is REAL.

An INTEGER is a whole number with no fractional part having a range from -32768 thru $+32767$.

Array Variable Names - can be any name that is appended with a subscript. Arrays are structured data types with 1 to 6 dimensions; each dimension is allowed 32767 elements. To specify an entire array, the characters (*) are placed after the array name. To specify a single element of an array, subscripts are placed in parentheses after the array name. If an array is not explicitly dimensioned, it is implicitly given the number of dimensions used in its first occurrence, with an upper bound of 10. The array variable names used in the following two listings are: Array(*), Iarray(*),Addr_array(*),Data_array(*),Channels(*),Destination(*),Range(*), and User(*). Iarray(*) is an array of INTEGER type values; all the others are of REAL type values. Note, single elements of array variables can be substituted for any numeric variable, e.g. Output(Name,Array[1,1]).

String Variable Names - String variable names are identical to numeric variable names with the exception of a dollar sign ($) added to to the end of the name. Message$ is an example of a string variable name. A string is a data type comprised of a contiguous series of ASCII characters. The default dimensioned length of a string is 18 characters. The DIM, COM, and ALLOCATE BASIC statements are used to define string lengths up to the maximum length of 32767 characters.

Literal - This is a string constant. When quote marks are used to delimit a literal, those quote marks are not part of the literal. To include a quote mark in a literal, type two consecutive quote marks. Examples of literal string expressions that appear in the following two listings are: "Menu_on", "Menu_off", "Ask_me", and "Filename".

Statement Summary (listing by function)

69700A-69706A Resistance Output Cards

Init(Name)
Output(Name, Value)

Advanced Programming Statements

Preset(Name,Value)
Start(Name1,...,Name8)
Wait_for(Name)
Check_done(Name,Flag)
Enable_intr(Name,Vector)
Disable_intr(Name)
Set_timeout(Name,Value)

69709A Power Supply Control Card

Init(Name)
Preset_current(Name,Value)
Output_voltage(Name,Value)

Advanced Programming Statements

Input(Name,Voltage_value,Current_value)
Preset_voltage(Name,Value)
Output_current(Name,Value)
Start(Name1,...,Name8)
Get_conditions(Name,Value)
Enable_intr(Name,Vector)
Disable_intr(Name)
Wait_for(Name)
Check_done(Name,Flag)
Enable_output(Name)
Disable_output(Name)
Set_ovp_trip(Name)
Reset_ovp_trip(Name)
Set_local_vprog(Name,Flag)
Set_local_iprog(Name,Flag)
Set_const_v_ien(Name,Flag)
Set_const_i_ien(Name,Flag)
Set_unreg_ien(Name,Flag)
Set_over_v_ien(Name,Flag)
Set_over_t_ien(Name,Flag)
Set_acfault_ien(Name,Flag)
Set_update_ien(Name,Flag)
Set_end_cyc_ien(Name,Flag)
FNConstvmode(Name)

69709A (continued)

FNConstimode(Name)
FNUnregout(Name)
FNOvptrip(Name)
FNOvertemp(Name)
FNAcfault(Name)
Set_timeout(Name,Value)

69720A/69721A Digital-to-Analog Converter Cards

Init(Name)
Output(Name,Value)

Advanced Programming Statements

Translate(Name,Iarray(•),User(•))
Output_iblock(Name,Iarray(•)[,Count[,Index]])
Preset(Name,Value)
Start(Name1,...,Name8)
Wait_for(Name)
Check_done(Name,Flag)
Enable_intr(Name,Vector)
Disable_intr(Name)
Set_timeout(Name,Value)

69730A Relay Output Card

Init(Name)
Output(Name,Value)

Advanced Programming Statements

Preset(Name,Value)
Start(Name1,...,Name8)
Wait_for(Name)
Check_done(Name,Flag)
Enable_intr(Name,Vector)
Disable_intr(Name)
Set_timeout(Name,Value)

69731B Digital Output Card

Init(Name)
Output(Name, Value)

Advanced Programming Statements

Translate(Name,Iarray(•),User(•))
Output_iblock(Name,Iarray(•)[,Count[,Index]])

69731B (continued)

Preset(Name,Value)
Start(Name1,...,Name8)
Wait_for(Name)
Check_done(Name,Flag)
Enable_intr(Name,Vector)
Disable_intr(Name)
Set_timeout(Name,Value)

69735A Pulse Train Output Card

Init(Name)
Set_period(Name,Value)
Output(Name,Value[,Period])
Stop(Name)

Advanced Programming Statements

Preset(Name,Value[,Period])
Start(Name1,...,Name8)
Wait_for(Name)
Check_done(Name,Flag)
Enable_intr(Name,Vector)
Disable_intr(Name)
Set_timeout(Name,Value)

69736A Timer/Pacer Card

Init(Name)
Set_oneshot(Name)
Set_continuous(Name)
Output(Name,Value)

Advanced Programming Statements

Preset(Name,Value)
Start(Name1,...,Name8)
Wait_for(Name)
Check_done(Name,Flag)
Enable_intr(Name,Vector)
Disable_intr(Name)
Set_timeout(Name,Value)

69750A Scan Control Card

Advanced Programming Statements

Init(Name)
Start(Name1,...,Name8)
Enable_intr(Name,Vector)
Disable_intr(Name)
Check_done(Name,Flag)
Wait_for(Name)

69750A (continued)

Set_switches(Name,Flag)
Enable_extrig(Name)
Disable_extrig(Name)
Set_singlepoint(Name)
Set_crosspoint(Name)
Set_stop_ien(Name,Flag)
Set_extlock_ien(Name,Flag)
Set_sequen_mode(Name)
Set_random_mode(Name)
Set_int_start(Name)
Set_ext_start(Name)
Set_int_pacing(Name)
Set_ext_pacing(Name)
Set_period(Name,Value)
Stop(Name)
Write_start_chn(Name,Value)
Write_stop_chnl(Name,Value)
Write_ext_chnl(Name)
Read_start_chnl(Name,Value)
Read_chnl_addr(Name,Value)
Read_stop_chnl(Name,Value)
Set_timeout(Name,Value)

69751A 33 kHz Analog-to-Digital Converter Card

Init(Name)
Input(Name,Value)

Advanced Programming Statements

Input_rblock(Name,Array(*)[,Count[,Index]])
Translate(Name,Iarray(*),User(*))
Input_iblock(Name,Iarray(*)[,Count[,Index]])
Start(Name1,...,Name8)
Get(Name,Value)
Wait_for(Name)
Check_done(Name,Flag)
Enable_intr(Name,Vector)
Disable_intr(Name)
Set_timeout(Name,Value)

69759A 500 kHz Analog-to-Digital Converter Card

Init(Name)
Input(Name,Value)

Advanced Programming Statements

Start(Name1,...,Name8)

69759A (continued)

Get(Name,Value)
Enable_intr(Name,Vector)
Disable_intr(Name)
Check_done(Name,Flag)
Wait_for(Name)
Input_rblock(Name,Array(•)[,Count[,Index]])
Input_iblock(Name,Iarray(•)[,Count[,Index]])
Translate(Name,Iarray(•),User(•)[,Range(•)])
Set_timeout(Name,Value)
Set_range(Name,Range)
Set_range_int(Name[,Range])
Set_range_ext(Name)
Enable_intrig(Name)
Disable_intrig(Name)
Enable_extrig(Name)
Disable_extrig(Name)
Enable_ext_tps(Name)
Set_extrig_lo(Name)
Set_extrig_hi(Name)
Set_extlock_lo(Name)
Set_extlock_hi(Name)
Disable_extlock(Name)
Set_single_trig(Name)
Set_multi_trig(Name)
Enable_mxctrl(Name)
Disable_mxctrl(Name)
Enable_output(Name)
Disable_output(Name)
Enable_een(Name)
Disable_een(Name)
Set_single_gate(Name)
Set_multi_gate(Name)
Set_sign_extn(Name)
Set_range_rdbk(Name)
Set_ext_xfer(Name)
Set_int_xfer(Name)

69770A Isolated Digital Input Card

Init(Name)
Input(Name,Value)

Advanced Programming Statements

Input_iblock(Name,Iarray(•)[,Count[,Index]])
Translate(Name,Iarray(•),User(•))
Get(Name,Value)
Start(Name1,...,Name8)
Wait_for(Name)

69770A (continued)

Check_done(Name,Flag)
Enable_intr(Name,Vector)
Disable_intr(Name)
Set_timeout(Name,Value)

69771A Digital Input/Analog Comparator Card

Init(Name)
Input(Name,Value)

Advanced Programming Statements

Input_iblock(Name,Iarray(•)[,Count[,Index]])
Translate(Name,Iarray(•),User(•))
Start(Name1,...,Name8)
Wait_for(Name)
Check_done(Name,Flag)
Get(Name,Value)
Enable_intr(Name,Vector)
Disable_intr(Name)
Set_timeout(Name,Value)

69775A Counter/Totalizer Card

Init(Name)
Input(Name,Value)

Advanced Programming Statements

Output(Name,Value)
Preset(Name,Value)
Start(Name1,...,Name8)
Wait_for(Name)
Check_done(Name,Flag)
Enable_intr(Name,Vector)
Disable_intr(Name)
Set_timeout(Name,Value)

69776A Interrupt Card

Init(Name)
Start(Name1,...,Name8)
Output(Name,Value)
Output_bit(Name,Bit,Value)
Write_mask(Name,Value)
Write_mask_bit(Name,Bit,Value)
Wait_for(Name)
Set_ne(Name)
Set_eq(Name)
Set_gt(Name)

69776A (continued)

Set_lt(Name)

Advanced Programming Statements

Preset(Name,Value)
Preset_bit(Name,Bit,Value)
Input(Name,Value)
Input_bit(Name,Bit,Value)
Read_intr_word(Name,Value)
Read_ref_word(Name,Value)
Check_done(Name,Flag)
Enable_intr(Name,Vector)
Disable_intr(Name)
Set_timeout(Name,Value)

69790B Memory Card

Init(Name)
Input_rblock(Name,Array(*)[,Count[,Index]])
Output_rblock(Name,Array(*)[,Count[,Index]])
Set_fifo_in(Name)
Set_fifo_out(Name)
Set_recirc_in(Name)
Set_recirc_out(Name)

Advanced Programming Statements

Input_iblock(Name,Iarray(*)[,Count[,Index]])
Output_iblock(Name,Iarray(*)[,Count[,Index]])
Translate(Name,Iarray(*),User(*))
Check_done(Name,Flag)
Wait_for(Name)
Read_rpointer(Name,Value)
Read_wpointer(Name,Value)
Read_ref_count(Name,Value)
Read_dif_count(Name,Value)
Write_rpointer(Name,Value)
Write_wpointer(Name,Value)
Write_ref_count(Name,Value)
Write_dif_count(Name,Value)
Enable_lockout(Name)
Disable_lockout(Name)
Enable_intr(Name,Vector)
Disable_intr(Name)
Set_timeout(Name,Value)

69791A/69792A Memory Card System

Init(Name)
Input_rblock(Name,Array(*)[,Count[,Index]])

69791A/69792A (continued)

Output_rblock(Name,Array(*)[,Count[,Index]])
Set_fifo_in(Name)
Set_fifo_out(Name)
Set_recirc_in(Name)
Set_recirc_out(Name)
Set_fifo_simul(Name)
Set_rec_simul(Name)

Advanced Programming Statements

Input_iblock(Name,Iarray(*)[,Count[,Index]])
Output_iblock(Name,Iarray(*)[,Count[,Index]])
Translate(Name,Iarray(*),User(*))
Check_done(Name,Flag)
Wait_for(Name)
Read_rpointer(Name,Value)
Read_wpointer(Name,Value)
Read_ref_reg1(Name,Value)
Read_ref_reg2(Name,Value)
Read_dif_count(Name,Value)
Write_rpointer(Name,Value)
Write_wpointer(Name,Value)
Write_ref_reg1(Name,Value)
Write_ref_reg2(Name,Value)
Write_dif_count(Name,Value)
Enable_lockout(Name)
Disable_lockout(Name)
Get_conditions(Name,Value)
Set_level_ien(Name,Flag)
Set_error_ien(Name,Flag)
Enable_intr(Name,Vector)
Disable_intr(Name)
Set_timeout(Name,Value)

3478A Digital Multimeter

Init(Name)
Input(Name,Value)

Advanced Programming Statements

Start(Name)
Get(Name,Value)
Enable_intr(Name,Vector)
Disable_intr(Name)
Check_done(Name,Flag)
Wait_for(Name)
Set_disp_mess(Name,Message$)
Set_display_on(Name)
Set_func_dcv(Name)

3478A (continued)

Set_func_acv(Name)
Set_func_dca(Name)
Set_func_aca(Name)
Set_func_2wo(Name)
Set_func_4wo(Name)
Set_func_exo(Name)
Set_range(Name,Range)
Set_rng_auto(Name)
Set_trigger_int(Name)
Set_trigger_ext(Name)
Set_digits_3(Name)
Set_digits_4(Name)
Set_digits_5(Name)
Set_autozero(Name,Flag)
Set_data_ien(Name,Flag)
Set_fr_srq_ien(Name,Flag)
Get_conditions(Name,Value)
Start_fast(Name)
Input_fast(Name,Value)
Set_timeout(Name,Value)

Buffer Function

Init(Name)
Input_rblock(Buffer_name,Array(•)
[,Count[,Index]])
Output_rblock(Buffer_name,
Array(•)[,Count[,Index]])
Set_fifo_in(Buffer_name)
Set_fifo_out(Buffer_name)
Set_recirc_in(Buffer_name)
Set_recirc_out(Buffer_name)

Advanced Programming Statements

Translate(Front_end_name,Iarray(•),
User(•)[,Range(•)])
Demultiplex(Buffer_name,Adc_name,
Iarray(•),Destination(•)[,Range(•)])

The remaining advanced programming statements for the Buffer function are the same as the advanced programming statements listed for the 69790B Memory card and the 69791A/69792A Memory Card System. Depending on the Memory Type choosen, either set of statements can be used. Remember to use the name of your buffer function for the Name parameter.

Scanner Function

Init(Name)
Load_channels(Name,Channels(•)[,Range(•)])
Start(Name1,...,Name8)

Advanced Programming Statements

The advanced programming statements for the Scanner function are the same as the advanced programming statements listed for the 69750A Scan Control card with the exception of Init(Name) and Start (Name1,...,Name8).

Timer Function

Init(Name)
Start(Name1,...,Name8)
Set_period(Name,Value)
Set_count(Name,Value)

Advanced Programming Statements

Wait_for(Name)
Check_done(Name,Value)
Enable_intr(Name,Vector)
Disable_intr(Name)
Stop(Name)
Set_burst_mode(Name)
Set_delay_mode(Name)
Set_reftrg_pos(Name)
Set_reftrg_neg(Name)
Set_timeout(Name,Value)

System Statements

System_init
Peek(Name,Subaddress,Value)
Poke(Name,Subaddress,Value)
Set_error_print(Flag)
Get_intr_vector(Vector,Reason,Select_code)
System_disable
System_enable
Test_Multi(Error_flag,Sc,[Bus_Addr])
Configure("Menu_on","Filename")
"Menu_off"
"Ask_me"

Statement Summary (alphabetic listing)

Check__done(Name,Flag) [5,6,7,8,9,10]

Flag is set to zero if the Named function has not completed an I/O operation after being started with a Start statement or external trigger signal.

Configure ("Menu__on","Filename"[,"Debug"])
"Menu__off"
"Ask__me" [3]

Either turns the configuration menu phase on or off, or prompts the operator for a decision. Filename is the Name of the file that the configuration data is stored in; it must be within quotes. Filename is a BDAT type file. The optional Debug parameter can be used for hardware verification purposes.

Demultiplex(Buffer__name,Adc__name,Iarray(*),
Destination(*)[,Range(*)]) [9]

Used after Input__iblock to process the data of one particular 69759A A/D card in multiplexed A/D applications. Iarray is the same array of integers used in the Input__iblock statement. Destination is the array of real numbers into which data is converted. The optional Range array parameter can be used when the range source of the A/D card is set to external.

Disable__een(Name) [5]

Prohibits the External Enable signal of the Named 69759A card from disabling the data outputs. Opposite of Enable__een(Name).

Disable__extlock(Name) [5]

Disables the external trigger lockout signal of the Named 69759A card.

Disable__extrig(Name) [5]

Disables the external trigger of a Named 69750A card, 69759A card, or Scanner function. Opposite of Enable__extrig(Name).

Disable__intr(Name) [5,6,7,8,9,10]

Disables any interrupts from Name. Opposite of Enable__intr(Name,Vector).

Disable__intrig(Name) [5]

Prohibits the Named 69759A card from responding to internal triggers. Opposite of Enable__intrig(Name).

Disable__lockout(Name) [9]

Turns off external trigger lockout of a Buffer or Memory function. Opposite of Enable__lockout(Name).

Disable__mxctrl(Name) [5]

Prohibits the multiplex circuitry of the Named 69759A card from disabling the data outputs. Opposite of Enable__mxctrl(Name).

Disable_output(Name) 6
Disables the output of the Named 69759A card or 69709A card. Opposite of Enable_output(Name).

Enable_een(Name) 5
Allows the External Enable signal of the Named 69759A card to disable the data outputs. Opposite of Disable_een(Name).

Enable_extrig(Name) 5
Enables the external trigger of a Named 69750A card, 69759A card, or Scanner Function. Opposite of Disable_extrig(Name).

Enable_ext_tps(Name) 5
Enables the trigger polarity select input of the Named 69759A card.

Enable_intr(Name,Vector) 5,6,7,8,9,10
Allows Named function to generate interrupts. Vector is a number, greater than 0, that you assign so that your User_handler subprogram can identify which function interrupted. Opposite of Disable_intr(Name).

Enable_intrig(Name) 5
Allows the Named 69759A card to respond to internal triggers. Opposite of Disable_intrig(Name).

Enable_lockout(Name) 9
Enables the external trigger lockout of a Buffer or Memory function. Opposite of Disable_lockout(Name).

Enable_mxctrl(Name) 5
Allows the CHAIN IN and CHAIN OUT signals of the multiplex control circuitry of the Named 69759A card to disable the data outputs. The outputs must be enabled and EEN must be true. Opposite of Disable_mxctrl(Name).

Enable_output(Name) 5,6
Enables the output of a Named 69759A card or 69709A card. Opposite of Disable_output(Name).

FNAcfault(Name) 6
Applicable only to the Named 69709A Power Supply Control card. Returns a value of 1 if an ac input line voltage fault has occurred; zero otherwise.

FNConstimode(Name) 6
Applicable only to the Named 69709A Power Supply Control card. Returns a value of 1 if the power supply is operating in the constant current (CC) mode; zero otherwise.

FNConstvmode(Name) 6

Applicable only to the Named 69709A Power Supply Control card. Returns a value of 1 if the power supply is operating in the constant voltage (CV) mode; zero otherwise.

FNOvertemp(Name) 6

Applicable only to the Named 69709A Power Supply Control card. Returns a value of 1 if the internal temperature of the power supply exceeds a safe level; zero otherwise.

FNOvptrip(Name) 6

Applicable only to the Named 69709A Power Supply Control card. Returns a value of 1 if the power supply's over voltage protection circuit has tripped; zero otherwise.

FNUnregout(Name) 6

Applicable only to the Named 69709A Power Supply Control card. Returns a value of 1 if a power supply regulation fault has been detected; zero otherwise.

Get(Name,Value) 5,7,10

Reads data (that has been stored by an external trigger signal, Start statement, or a previously executed Input statement) from the Named input function and puts it into the variable "Value".

Get_conditions(Name,Value) 5,6,9

Returns the status of a Named 69709A card, 3478A Multimeter, or 69791A/69792A Memory Card System in the variable Value as follows:

69709A condition	3478A condition	69791A/69792A condition	Value
Constant Voltage mode	Data Ready	EOP is set	1
Constant Current mode	Name not currently active	Card is busy	2
Unregulated Output	Syntax Error	Error interrupts masked	4
Over Voltage Trip	Hardware Error	Write to full memory	8
Over Temperature Trip	Front Panel SRQ	Read from empty memory	16
Input Supply Fault	Cal Failed	Read cycle in progress	32
	SRQ	Write cycle in progress	64
	Power-on Reset		128

Get_intr_vector(Vector,Reason,Select_code) 10

Returns the Vector, Reason and Select_code of the function which causes an interrupt. Vector was assigned by you in your Enable_intr(Name,Vector) statement to allow you to determine which function interrupted. Reason allows you to determine which of several conditions caused a specific function to interrupt. Reason is undefined except for the following five functions:

Interrupt Reason	69776A	69709A	69750A	3478A	69791A/ 69792A
1	Bit 0	Constant Voltage mode	Stop Channel Reached	Data Available	Level
2	Bit 1	Constant Current mode	External Lockout	Front Panel SRQ	Missed Data
4	Bit 2	Unregulated Output			Invalid Read
8	Bit 3	Over Voltage Trip			
16	Bit 4	Over Temperature Trip			
32	Bit 5	Input Supply Fault			
64	Bit 6	Readback Update			
128	Bit 7	End Of Cycle			
256	Bit 8				
512	Bit 9				
1024	Bit 10				
2048	Bit 11				
4096	Bit 12				
8192	Bit 13				
16384	Bit 14				
−32768	Bit 15				

Select_code tells you on which interface the interrupt occurred, providing that its value is greater than 0. A Select_code of 0 indicates a spurious interrupt; e.g. the interrupting device was not found. A Select_code of less than 0 indicates that a Multiprogrammer interrupted, but the requesting card was not found.

Init(Name) 3,5,6,7,8,9

Sets Name to the initial conditions that were chosen in the configuration menu and disables interrupts from Name. Functions can not be programmed until they are initialized.

Input(Name,Value) 3,5,7,9,10

Causes the Named function to take a reading, and returns it into the variable Value.

Input(Name,Voltage_value,Current_value) 6

Reads the values of Voltage and Current from the Named 69709A card.

Input_bit(Name,Bit,Value) 9

Reads the current value of the specified bit of the Named 69776A Interrupt Card's input word. Bit can range from 0 to 15 minus Start Bit. Value is returned as either 0 or 1.

Input_fast(Name,Value) 5

Causes the Named 3478A to take a reading without allowing the usual settling time for the input filters, and returns the reading in the variable Value.

Input_iblock(Name,Iarray(•)[,Count[,Index]]) 5,7,9

Reads unscaled integer data from the Named Buffer function, Digital Input function, Analog Input function, or Memory function and puts it into the integer array, Iarray (•). Count is the number of integers that you want to read from Name. Index is the starting position in Iarray(•) where data is to be placed. Iarray(•) must be DIMensioned or ALLOCATEd.

Input_rblock(Name,Array(*)[,Count[,Index]]) 5,9

Reads data from the Named Buffer function, Analog Input function, or Memory function and puts it into the real array, Array(*). The data may be scaled by User Conversion, if desired. Count is the number of data words (type REAL) that you want to read from Name. Index is the starting position in Array(*) where data is to be placed. Array(*) must be DIMensioned or ALLOCATEd.

Load_Channels(Name,Channels(*)[,Range(*)]) 5

Loads the external channel memory specified in the Named Scanner function from the Channels array. Can be used only if the Scan Mode is Random. Channels is the array that contains the previously entered channels to be scanned. The optional Range array parameter can be used if the Scanner function has been referenced from the 69759A card.

Output(Name,Count[,Period]) 9

Sends a Count value to the Named 69735A card and then starts it. Period is an optional parameter that specifies the period of the pulse that's generated.

Output(Name,Value) 3,6,8,9,10

Sends data Value to the Named function then Starts it. The Output statement will not finish until the function has completed its operation (with the exception of the 69776A). Value may be scaled by User Conversion, if desired.

Output_bit(Name,Bit,Value) 9

Sets the specified Bit in the Reference Word of the Named 69776A Interrupt Card to Value, and then Starts the function. Bit can range from 0 to 15 minus Start Bit. Value can be either 0 or 1.

Output_current(Name,Value) 6

Sends a Value of current to the Named 69709A Power Supply Control card, and programs the power supply with this current at the preset voltage. If output sampling has been selected, current is the parameter sampled.

Output_iblock(Name,Iarray(*)[,Count[,Index]]) 6,8,9

Sends unscaled data to the Named Buffer function, Digital Output function, Analog Output function, or Memory function from Iarray(*). Index is the starting position of Iarray(*) from where data will be sent. Iarray(*) must be DIMensioned or ALLOCATEd.

Output_rblock(Name,Array(*)[,Count[,Index]]) 6,9

Sends a read data to the Named Buffer function, or Memory function from Array(*). The data may be scaled by User Conversion, if desired. Count is the number of data words that are to be sent from Array(*). Index is the starting position of Array(*) from where data will be sent. Array(*) must be DIMensioned or ALLOCATEd.

Output_voltage(Name,Value) 6

Sends a voltage Value to the Named 69709A Power Supply Control card and programs the power supply with this voltage at the preset current. If output sampling has been selected, voltage is the parameter sampled.

Peek(Name,Subaddress,Value) [10]

Reads the Named card's Subaddress (0 through 3) and places the data in Value. It does not cycle the card (to cycle the card send the Start statement). Its use is limited to single-card functions with the exception of the 69790B Memory card and the 69791A/69792A Memory Card System. The values of Subaddress that are used to access Memory card #2 of the 69790B are 4 through 7.

Poke(Name,Subaddress,Value) [10]

Sends the Named card's Subaddress (0 through 3) the data specified in Value. It does not cycle the card (to cycle the card use the Start statement). Its use is limited to single-card functions with the exception of the 69790B Memory card and the 69791A/69792A Memory Card System. The values of Subaddress that are used to access Memory card #2 of the 69790B are 4 through 7.

Preset(Name,Count[,Period]) [9]

Count and optional Period are loaded into the Named 69735A card but the card is not Started.

Preset(Name,Value) [6,8,9,10]

Value is loaded into the Named function; the card is not Started. Value may be scaled by User conversion, if desired.

Preset_bit(Name,Bit,Value) [9]

Sets the specified Bit in the Reference Word of the 69776A Interrupt card to Value, but the function is not Started. Bit can range from 0 to 15 minus Start Bit. Value can be either 0 or 1.

Preset_current(Name,Value) [6]

Data Value of current is loaded into the Named 69709A card but the new value is not sent to the power supply. The output sampling parameter is not affected.

Preset_voltage(Name,Value) [6]

Data Value of voltage is loaded into the Named 69709A card but the new value is not sent to the power supply. The output sampling parameter is not affected.

Read_chnl_addr(Name,Value) [5]

The present channel address of the Named 69750A card or Scanner function is returned in Value.

Read_dif_count(Name,Value) [9]

The difference count of the Named Buffer or Memory function is returned in Value.

Read_intr_word(Name,Value) [9]

The interrupting word of the Named 69776A Word Interrupt card is returned in Value.

Read_ref_count(Name,Value) [9]

Applies only when using the 69790B Memory card. Returns the reference count of the Named Buffer or Memory function in Value.

Read_ref_reg1(Name,Value) 9
Applies only when using the 69791A/69792A Memory Card System. Returns the contents of reference register 1 of the Named Buffer or Memory function in Value.

Read_ref_reg2(name,Value) 9
Applies only when using the 69791A/69792A Memory Card System. Returns the contents of reference register 2 of the Named Buffer or Memory function in Value.

Read_ref_word(Name,Value) 9
Returns the current Reference Word of the Named 69776A Interrupt card in Value.

Read_rpointer(Name,Value) 9
Returns the read pointer address of the Named Buffer or Memory function in Value.

Read_start_chnl(Name,Value) 5
Returns the start channel address of the Named 69750A card or Scanner function in Value.

Read_stop_chnl(Name,Value) 5
Returns the stop channel address of the Named 69750A card or Scanner function in Value.

Read_wpointer(Name,Value) 9
Returns the write pointer address of the Named Buffer or Memory function in Value.

Reset_ovp_trip(Name) 6
Resets the over voltage protection circuit of the Named 69709A card. Opposite of Set_ovp_trip(Name).

Set_acfault_ien(Name,Flag) 6
Allows the Named 69709A card to interrupt the computer if the controlled power supply has detected an ac line voltage fault. For an interrupt to occur an Enable_intr statement must be programmed. In addition, the BASIC ON INTR and 14752A Enable_sc_intr statements must be in effect. If Flag = 0, this condition will not generate an interrupt.

Set_autozero(Name,Flag) 5
Places the Named 3478A Multimeter in autozero mode if Flag = 1. Turns off autozero mode if Flag = 0.

Set_burst_mode(Name) 9
Applies only when using a 69731B Digital Output card. Programs the Named Timer function to burst mode.

Set_const_i_ien(Name,Flag) 6
If Flag = 1, allows the Named 69709A card to interrupt the computer when the controlled power supply goes into constant current mode. For an interrupt to occur an Enable_intr statement must be programmed. In addition, the BASIC ON INTR and 14752A Enable_sc_intr statements must be in effect. If Flag = 0, this condition will not generate an interrupt.

Set_const_v_ien(Name,Flag) 6
If Flag = 1, enables the Named 69709A card to interrupt the computer when the controlled power supply goes into constant voltage mode. For an interrupt to occur, and Enable_intr statement must be programmed. In addition, the BASIC ON INTR and 14752A Enable_sc_intr statements must be in effect. If Flag = 0, this condition will not generate an interrupt.

Set_continuous(Name) 9
The Named 69736A Timer/Pacer card is set into continuous output mode. Opposite of Set_oneshot(Name).

Set_count(Name,Value) 9
Sets the number of output pulses before the Named Timer function completes.

Set_crosspoint(Name) 5
Turns on the crosspoint enable feature of the Named 69750A card or Scanner function. Opposite of Set_singlepoint(Name).

Set_data_ien(Name,Flag) 5
If Flag = 1, allows the Named 3478A to interrupt the computer when a reading is completed. For an interrupt to occur an Enable_intr statement must be programmed. In addition, the BASIC ON INTR and ENABLE INTR statements must be in effect. If Flag = 0, this condition will not generate an interrupt.

Set_delay_mode(Name) 9
Applies only when using a 69731B Digital Output card. Programs the Named Timer function to delay mode.

Set_digits_3(Name) 5
Sets the display of the Named 3478A Multimeter to 3½ digits.

Set_digits_4(Name) 5
Sets the display of the Named 3478A Multimeter to 4½ digits.

Set_digits_5(Name) 5
Sets the display of the Named 3478A Multimeter to 5½ digits.

Set_display_on(Name) 5
Clears the display message and returns the Named 3478A display to its normal mode. Refer to the Set_disp_mess description below.

Set_disp_mess(Name,Message$) 5
Sends a message contained in "String" to the display of the Named 3478A display. String must be ASCII and can be up to 12 letters long. See Set_display_on(Name).

Set_end_cyc_ien(Name,Flag) 6
If Flag = 1, allows the Named 69709A card to interrupt the computer if the card has finished sending data to the controlled power supply. For an interrupt to occur an Enable_intr statement must be programmed. In addition, the BASIC ON INTR and 14752A Enable_sc_intr statements must be in effect. If Flag = 0, this condition will not generate an interrupt.

Set__eq(Name) 9
Programs the Named 69776A Word Interrupt card to the Equal comparison mode.

Set__error__ien(Name,Value) 9
Applies only when using the 69791A/69792A Memory Card System. Sets error interrupts of the Named Buffer or Memory function. If Flag = 1, error interrupts will be enabled by the Enable__intr statement. If Flag = 0, error interrupts are disabled by the Enable__intr statement.

Set__error__print (Flag) 4
System command that suppresses the printing of 14752A error messages if Flag = 0 and allows the printing of error messages if Flag = 1. Default is message printing enabled; you would suppress them if you wish to handle errors with your own ON ERROR routine.

Set__ext__pacing(Name) 5
Allows the Named 69750A card or Scanner function to be paced by an external source as opposed to using its internal pacer. Opposite of Set__int__pacing(Name).

Set__ext__start(Name) 5
Allows the start channel of the Named 69750A card or Scanner function to be supplied by external hardware. Opposite of Set__int__start(Name).

Set__ext__xfer(Name) 5
Prepares the Named 69759A card for an Input__iblock or Input__rblock with external pacing. Opposite of Set__int__xfer(Name).

Set__extlock__hi(Name) 5
Prevents external triggering when the External Trigger Lockout signal of the Named 69759A card is high. Opposite of Set__extlock__lo(Name).

Set__extlock__ien(Name,Flag) 5
If Flag = 1, allows the Named 69750A card or Scanner function to interrupt the computer if lockout is enabled externally. For an interrupt to occur an Enable__intr statement must be programmed, In addition, the BASIC ON INTR and 14752A Enable__sc__intr statements must be in effect. If Flag = 0, this condition will not generate an interrupt.

Set__extlock__lo(Name) 5
Prevents external triggering when the External Trigger Lockout signal of the Named 69759A card is low. Opposite of Set__extlock__hi(Name).

Set__extrig__hi(Name) 5
Sets the external trigger signal of the Named 69759A card to be active high.

Set__extrig__lo(Name) 5
Sets the external trigger signal of the Named 69759A card to be active low.

Set__fifo__in(Name) 9
Programs the Named Buffer or Memory function into FIFO input mode.

Set_fifo_out(Name) 9
Programs the Named Buffer or Memory function into FIFO output mode.

Set_fifo_simul(Name) 9
Sets the Named 69791A/69792A Memory Card System's operating mode to the Simultaneous FIFO Input mode.

Set_fp_srq_ien(Name,Flag) 5
If Flag = 1, allows the front panel service request button of the Named 3478A to interrupt the computer when a reading is completed. For an interrupt to occur an Enable_intr statement must be programmed. In addition, the BASIC ON INTR and ENABLE INTR statements must be in effect. If Flag = 0, this condition will not generate an interrupt.

Set_func_aca(Name) 5
Programs the Named 3478A Multimeter to read ac amperes.

Set_func_acv(Name) 5
Programs the Named 3478A Multimeter to read ac volts.

Set_func_dca(Name) 5
Programs the Named 3478A Multimeter to read dc amperes.

Set_func_dcv(Name) 5
Programs the Named 3478A Multimeter to read dc volts.

Set_func_exo(Name) 5
Selects the Extended Ohms function of the Named 3478A.

Set_func_2wo(Name) 5
Selects the two wire ohms mode of the Named 3478A.

Set_func_4wo(Name) 5
Selects the four wire ohms mode of the Named 3478A.

Set_gt(Name) 9
Sets the Named 69776A Word Interrupt card to the Greater Than comparison mode.

Set_int_pacing(Name) 5
Allows the Named 69750A card or Scanner function to be paced by the on-board Pacer as opposed to using an external source. Opposite of Set_ext_pacing(Name).

Set_int_start(Name) 5
Allows the start channel address of the Named 69750A card or Scanner function to be supplied by the program. Opposite of Set_ext_start(Name).

Set_int_xfer(Name) 5
Prepares the Named 69759A card for an Input_iblock or Input_rblock with no external pacing. Opposite of Set_ext_xfer(Name).

Set_level_ien(Name,Flag) 9
Applies only when using the 69791A/69792A Memory Card System. Sets level interrupts of the Named Buffer or Memory function. If Flag = 1, level interrupts will be enabled by the Enable_intr statement. If Flag = 0, level interrupts are disabled by the Enable_intr statement.

Set_local_iprog(Name,Flag) 6
Applies only to the Named 69709A Power Supply Control card. Enables local (front panel) control of the power supply output current when Flag = 1. Switches control of the power supply output current to remote if Flag = 0.

Set_local_vprog(Name,Flag) 6
Applies only to the Named 69709A Power Supply Control card. Enables local (front panel) control of the power supply output voltage when Flag = 1. Switches control of the power supply output voltage to remote if Flag = 0.

Set_lt(Name) 9
Programs the Named 69776A Word Interrupt card to the Less Than comparision mode.

Set_multi_gate(Name) 5
When in multiple gate mode, allows a Start, Input_iblock, or Input_rblock to cause the Gate signal of the Named 69759A card to go active until reset by the EOC signal.

Set_multi_trig(Name) 5
Allows the Named 69759A card to respond to multiple triggers (internal or external).

Set_ne(Name) 9
Programs the Named 69776A Word Interrupt card to the Not Equal comparison mode.

Set_oneshot(Name) 9
Places the Named 69736A Timer/Pacer card in the Oneshot mode. Opposite of Set_continuous(Name).

Set_over_t_ien(Name,Flag) 6
If Flag = 1, enables the Named 69709A card to interrupt the computer if the internal temperature of the controlled power supply exceeds a safe level. For an interrupt to occur, and Enable_intr statement must be programmed. In addition, the BASIC ON INTR and 14752A Enable_sc_intr statements must be in effect. If Flag = 0, this condition will not generate an interrupt.

Set_over_v_ien(Name,Flag) 6
If Flag = 1, allows the Named 69709A card to interrupt the computer if the Enable_intr statement has already been issued, Flag = 1, and power supply goes into over-voltage mode. For an interrupt to occur, and Enable_intr statement must be programmed. In addition, the BASIC ON INTR and 14752A. Enable_sc_intr statements must be in effect. If Flag = 0, this condition will not generate an interrupt.

Set_ovp_trip(Name) 6
Programatically sets the over-voltage protection circuits of the Named power supply, thereby forcing the power supply output to zero volts and zero current. Opposite of Reset_ovp_trip(Name).

Set__period(Name,Value) [5,9]
Tells the Named 69735A card, 69750A card, Timer function, or Scanner function what period the output pulse must be.

Set__random__mode(Name) [5]
Programs the Named 69750A card to the random mode. Opposite of Set__sequen__mode(Name).

Set__range(Name,Range) [5]
Sets the range of the Named 3478A Multimeter and the Named 69759A card. The range of the 3478A must be one of the following: .030, .300, 3, 30, 300, 3 k, 30 k, 300 k, 3 M, or 30 M. The internal input range of the 69759A must be within 2.5% of one of the following nominal input ranges: 5, .5, .05, 50, 10, 1, .1, 100.

Set__range__ext(Name) [5]
Allows the range of the Named 69759A card to be supplied externally at the card edge.

Set__range__int(Name[,Range]) [5]
Allows the range of the Named 69759A card to be supplied internally. The value specified by the previous Set__range command is used unless the optional Range parameter is specified.

Set__range__rdbk(Name) [5]
Specifies that the returned data of the Named 69759A card will have range information. Must be set when using Get or Input__iblock with externally supplied range.

Set__rec__simul(Name) [9]
Sets the Named 69791A/69792A Memory Card System's operating mode to the Simultaneous Recirculate Input mode.

Set__recirc__in(Name) [9]
Programs the Named Buffer or Memory function to the Recirculate input mode.

Set__recirc__out(Name) [9]
Programs the Named Buffer or Memory function to the Recirculate output mode.

Set__reftrig__neg(Name) [9]
Applies only when using a 69731B Digital Output card. Makes the logic sense of the reference trigger in a Named Timer function negative true. Opposite of Set__reftrig__pos(Name).

Set__retrig__pos(Name) [9]
Applies only when using a 69731B Digital Output card. Makes the logic sense of the reference trigger in a Named Timer function positive true. Opposite of Set__reftrig__neg(Name).

Set__rng__auto(Name) [5]
Selects the autorange mode of the Named 3478A Multimeter.

Set__sequen__mode(Name) [5]
Programs the Named 69750A card or Scanner function to the sequential mode. Opposite of Set__random__mode(Name).

Set_sign_extn(Name) 5
Allows data from the Named 69759A card to be returned sign extended. This provides a 16-bit integer from a 12-bit A/D card reading.

Set_single_gate(Name) 5
When in single gate mode, allows a Start, Input_iblock, or Input_rblock to cause the Gate signal of the Named 69759A card to go active and remain so.

Set_single_trig(Name) 5
Allows the Named 69759A card to respond only to the first trigger (internal or external).

Set_singlepoint(Name) 5
Returns the Named 69750A card or Scanner function to its normal operating mode that allows no more than one switch on any of the Scanner cards to be closed at one time. This statement is the opposite of Set_crosspoint.

Set_stop_ien(Name,Flag) 5
If Flag = 1, Allows the Named 69750A card or Scanner function to interrupt the computer if the present Scanner channel address is equal to the Stop channel address. For an interrupt to occur, an Enable_intr statement must be programmed. In addition, the BASIC ON INTR and 14752A Enable_sc_intr statements must be in effect. If Flag = 0, this condition will not generate an interrupt.

Set_switches(Name,Flag) 5
Applies only to the Named 69750A Scan Control card or Scanner function. If Flag = 0, any selected Scanner switches are forced open. Selecting any subsequent switches will cause them to be remembered, but not closed. If Flag = 1, any presently selected switches will be closed.

Set_timeout(Name,Value) 5,6,7,8,9
Sets the maximum time that will be allowed for Name to complete certain operations (usually Output, Input, and Wait_for). Value is in seconds.

Set_trigger_ext(Name) 5
Allows the use of the External Trigger signal on the Named 3478A function. Opposite of Set_trigger_int(Name).

Set_trigger_int(Name) 5
Locks out the External Trigger signal on the Named 3478A function. Opposite of Set_trigger_ext(Name).

Set_unreg_ien(Name,Flag) 6
If Flag = 1, allows the Named 69709A card to interrupt the computer if the controlled power supply has gone out of regulation. For an interrupt to occur an Enable_intr statement must be programmed. In addition, the BASIC ON INTR and 14752A Enable_sc_intr statements must be in effect. If Flag = 0, this condition will not generate an interrupt.

Set_update_ien(Name,Flag) [6]

If Flag = 1, allows the Named 69709A card to interrupt the computer if either the output voltage or output current readings are updated. This will occur typically every 100 milliseconds. For an interrupt to occur an Enable_intr statement must be programmed. In addition, the BASIC ON INTR and 14752A Enable_sc_intr statements must be in effect. If Flag = 0, this condition will not generate an interrupt.

Start(Name1,...,Name8) [5,6,7,8,9,10]

Causes the Named I/O cards to perform their main I/O function. There can be up to eight Names in the Start statement; all the Names must be at the same select code.

Start_fast(Name) [5]

Causes the Named 3478A to take a reading without allowing the usual settling time for the input filters.

Stop(Name) [5,9]

Stops the pulse train of the Named 69750A card, 69735A card, Timer function or Scanner function.

System_disable [4]

Disables the outputs of all output cards in all Multiprogrammers that contained at least one I/O function that was configured. Opposite of System_enable.

System_enable [4]

Enables the main output of all output cards in all Multiprogrammers that contained at least one I/O function that was configured. Opposite of System_disable.

System_init [3,4]

Causes all Named functions in the system to return to the initial conditions that were determined in the configuration stage (menus). Also disables all interrupts. Named functions are initialized in the order in which they appear in the COMmon statement.

Test_multi(Error_flag,Sc,[Bus_Addr]) [3]

An optional statement used to selftest the Multiprogrammer. Invokes a selftest on the 6942A when Error_flag, Sc, and Bus_Addr are specified and on the 6944A when Error_flag, and Sc are specified. If the selftest fails, the Error_flag returns the number of errors detected otherwise a zero is returned. This statement should be used before the configure statement.

Translate(Name,Iarray(*),User(*)) [5,6,7,8,9]

Used in conjunction with Input_iblock for input cards and Output_iblock for output cards to process card data. Iarray is the same array of integers used in the Input_iblock and Output_iblock statements. User is an array of real numbers.

Translate(Name,Iarray(*),User(*)[,Range(*)]) [5,9]

Used in conjunction with Input_iblock for the Named 69759A A/D card or Buffer function (with a 69759A front-end card). The optional Range parameter can be used to process the results if the range source of the 69759A is set to external.

Wait_for(Name) [5,6,7,8,9]

After a function is started with a Start statement or external trigger signal, this statement will suspend your program until the Named function either completes its operation or a timeout occurs.

Write_dif_count(Name Value) 9
Programs the differential count of the Named Buffer or Memory function to Value.

Write_ext_chnl(Name) 5
Sets the channel address source of the Named 69750A card or Scanner function to external and selects the channel via the external address lines (IN00-IN09). Used to select channels in random mode or specify the start channel in sequential mode. Opposite of Write_start_chnl(Name,Value).

Write_mask(Name,Value) 9
Sets the mask word of the Named 69776A Word Interrupt card to Value.

Write_mask_bit(Name,Bit,Value) 9
Sets the specified bit of the Mask Word of the Named 69776A Interrupt card to Value. Bit can range from 0 to 15 minus Start Bit. A value of 1 enables the corresponding bit to figure in the comparison; a Value of 0 disables that bit from the comparison.

Write_ref_count(Name,Value) 9
Applies only when using the 69790B Memory card. Sets the reference count of the Named Buffer or Memory function to Value.

Write_ref_reg1(Name,Value) 9
Applies only when using the 69791A/69792A Memory Card System. Sets reference register 1 for the Named Buffer or Memory function to Value.

Write_ref_reg2(Name,Value) 9
Applies only when using the 69791A/69792A Memory Card System. Sets reference register 2 for the Named Buffer or Memory function to Value.

Write_rpointer(Name,Value) 9
Sets the read pointer of the Named Buffer or Memory function to Value.

Write_start_chn(Name,Value) 5
Sets the channel address source of the Named 69750A card or Scanner function to internal and selects the start channel address specified by Value. Used to select channels in the random mode or specify the start channel in the sequential mode. Opposite of Write_ext_chnl(Name).

Write_stop_chnl(Name,Value) 5
Sets the stop channel for the Named 69750A card or Scanner function to Value.

Write_wpointer(Name,Value) 9
Sets the write pointer of the Named Buffer or Memory function to Value.

Appendix A
Multiprogrammer Verification Library

Introduction

Included on the 14752A UTILITY PROGRAMS DISC are verification/troubleshooting programs provided to help you isolate system problems to software or hardware origins. For each 14752A single I/O card function (excluding the 69709A Power Supply Control Card) a "user interactive" program is available to exercise the primary function of that I/O card. Calibration routines are provided for the 69751A 33 kHz A/D and the 69759A 500 kHz A/D as part of the verification programs for those cards. Some additional information on the verification subprogram Test__multi is located at the rear of this appendix.

Verification Hardware

The hardware necessary to perform functional verification of I/O cards is minimal. The I/O Card Edge Connector Test Assembly which is included in the 14752A Package, is used to provide convenient connection points to an I/O card for oscilloscope and DVM probes when monitoring the performance of an I/O card during testing. Unpack and assemble the Edge Connector Test Assembly using the instruction sheet contained in the package.

Because Multiprogrammer systems are capable of stimulus, acquisition, and digital signals, it is necessary that several types of test equipment be available for use in verifying the operation of individual I/O cards. Equipment necessary to perform functional verification of all types of I/O cards include the following:

1. An oscilloscope (bandwidth of 20 MHz or greater)
2. A Logic Probe
3. A Digital Multimeter with resistance, voltage, and current measuring capabilities

The test equipment required for any specific test depends on the particular I/O card to be verified. Consult the test specifics for the I/O cards given in the "Individual Tests" portion of this appendix for the exact requirements.

In addition, several clip leads or wires (at least 10 inches long) are required.

Calibration Hardware

The A/D calibration routines require a precision Voltage Source, precision DVM and a 20 V peak to peak square wave source (for the 69759A only) to be used during the procedures. Required specifications for the voltage source are ± 100 V with .005% accuracy. The DVM should be 5½ digits with .005% accuracy. Consult Section 5 of the appropriate I/O card operating manual for complete details on the test equipment, including recommended models.

Software

Loading the Verification Library

The functional test programs are accessed by loading and running the BASIC program named "VERIFY" which is located on the 14752A UTILITY PROGRAMS disc. This program can be used with either the BASIC 2.1, BASIC 3.0 or BASIC 4.0 operating system.

Note that an operating system (Either BASIC 2.1, BASIC 3.0 or BASIC 4.0) must be loaded into the computer prior to loading and running "VERIFY". In addition, for Model 236 Color computers, the graphics binary "GRAPH2__1" must also be loaded.

Complete details on loading your operating system are found in "Setting up the System Software", Chapter 2. For those using the BASIC 3.0 and BASIC 4.0 operating system, all of the Driver and Language BIN files specified in Chapter 2 are required to run the verification software.

To load "VERIFY", place the 14752A Utility disc into the default disc drive of the computer you are using , then type:

LOAD "VERIFY"
and press EXECUTE or RETURN

Next, remove the Utility Programs disc from the disc drive and insert the appropriate 14752A CAT Programming Package disc based on your version of the BASIC operating system.

Load the 14752A Library by typing:

LOADSUB ALL FROM "LIBRARY__4" (for 4.0 systems)
LOADSUB ALL FROM "LIBRARY__3" (for 3.0 systems)
or
LOADSUB ALL FROM "LIBRARY__2" (for 2.1 systems)
and press EXECUTE or RETURN

Running the Verification Programs

You are now ready to run the verification programs; press RUN. After a few seconds (required by the computer to perform pre-run) the screen will display "14752A SYSTEM VERIFICATION PROGRAMS" and present the choice of two softkeys; "HELP" and "CONTINUE". The "HELP" softkey displays a condensed version of the configuration instructions contained in the next section and will reference this appendix for further information. The "CONTINUE" softkey causes the program to advance directly to the configuration procedure.

If you have left the disc in the default drive after you loaded the 14752A Library, then just press the "CONTINUE" key. If you have already removed the disc, then reinsert the disc into the drive, close the drive door, and then press "CONTINUE". The program proceeds to make sure that you have not forgotten to load the 14752A Library. If the 14752A Library has not been loaded into the computer for some reason, the following message is displayed and the program stops:

"LIBRARY" HAS NOT BEEN LOADED
INSERT THE 14752A CAT PROGRAMMING
PACKAGE DISC, AND TYPE:

LOADSUB ALL FROM "LIBRARY_3"
THEN PRESS "ENTER" AND RUN THIS PROGRAM AGAIN.

If the library has been loaded into the computer, then the program makes sure that the 14752A CAT Programming Package disc is present in the Default disc drive and that the drive door is closed. If the program finds that the Default disc drive is open or the wrong disc is present in the disc drive, then prompts such as "Insert the 14752A CAT DISC" or "Close the drive door" are displayed until the situation has been corrected and the "CONTINUE" softkey is pressed.

Configuring

"VERIFY" requires you to create a configuration file called "TEST" and stores it on the 14752A CAT Programming Package disc. If this is the first time verification is being performed, then you must proceed through the entire procedure below. However, if this is not the first time, then several options are available to you. If the file "TEST" is contained on the disc and is correct (i.e., all cards to be tested are configured properly) then answer "NO" by pressing the appropriate softkey when the question "Go thru configuration menus?" is displayed. If the file "TEST" already exists but is incomplete or does not represent the contents of the Multiprogrammer at this time, then proceed with the configuration sequence but change only what is necessary. Remember, if you have configured an I/O card in slot X and, for some reason it has been removed, a 14752A error will be generated. If it is necessary to configure or to reconfigure, then answer "YES" by pressing this appropriate softkey when the question "Go thru configuration menus?" is displayed. If the file "TEST" is not located on the disc (either because you have the wrong disc inserted into the default disc drive or this is the first time verification is being performed on the system), then the program informs you that "TEST" is not present. Just press ENTER and "TEST" will be created. Now enter the year, month, etc. in the first menu. Then press the "START" softkey.

The choice of names presented is "Slot0, Slot1, Slot2,...Slot15" which matches the layout of a Multiprogrammer Mainframe. Select the "name" corresponding to the slot in which the I/O card to be verified is located. Now select the type of I/O card function to be tested. Do not select any multiple I/O card functions, the 69709A Power Supply Programming card, or the 3478A DMM. Just select from the other single card functions (See note). Now enter the correct slot number, Multiprogrammer Type Interface Select Code, HP-IB Address (if necessary) and Frame Number. The slot number you enter should be the same as the "name" you have selected; i.e., the function named Slot0 should be located in slot 0 of the Multiprogrammer.

NOTE

To test a multiple I/O card function, test each one of the cards individually using the appropriate functional test.

Never change any of the menu parameters of the I/O cards you are testing with the exception of the 69791A Memory Systems, 69751A and 69759A A/D card and the 69700A Custom Resistance cards (see associated write-ups later in the appendix). The functional test programs were written using the default values and formats given in the menu. Changing any of these parameters may cause the test to fail. The philosophy of performing verification is to eliminate as many of the unknows as possible. By configuring the function to be tested with default values, the test is known to work if the I/O card is good and no switches or jumpers on the card have been changed from the factory settings. Now press "Next Name" and configure any other I/O cards to be tested.

When you have finished configuring all the functions to be tested, select the "End Menu" softkey and then store or restore the file using softkeys as "STORE" or "RESTORE" then "ENTER". Finally, exit the configuration procedure using the "END Menu" softkey again.

A special form of Configure is used with the verification programs. By adding an optional third parameter to the line which calls Configure, it is possible to program a known faulty I/O card or Multiprogrammer for verification or troubleshooting purposes. Adding the literal string "DEBUG" to the call to Configure (as shown below) changes the way in which detected failures are handled.

Configure ("ASK ME", "TEST", "DEBUG")

Normally any Multiprogrammers, Interfaces, or I/O cards specified during configurations are tested prior to exiting Configure. If a card or interface fails these tests you cannot exit Configure. Thus you cannot test or troubleshoot a system problem. By specifying the "DEBUG" parameter when calling Configure, error messages concerning the first system failure are still printed, but further testing of the system stops and Configure ends.

Thus communication with faulty I/O cards for troubleshooting purposes is possible. There is the added benefit of not having to remove any faulty I/O cards from the system before you can operate the remainder of the hardware.

One consideration in using "DEGUG" is that, if one of the verification tests does fail, System__enable is not set true. The System__enable command must be used prior to programming any output cards. (This is done automatically by the verification program.)

System_selftest

The program now performs a comprehensive test on all Multiprogrammers, Interfaces, and I/O cards specified in the configuration file "TEST". Full system status information is displayed on the screen.

For each 98633A Interface identified in the configuration file, the screen will display:

98633A Interface at Select Code XX
followed by either a pass or fail message

If the interface passes then all 6944A Multiprogrammers connected to it are tested slot by slot to determine if there are any I/O card errors. Status information for each slot tested is displayed.

For specified 6942A/6943A systems, the program causes the Multiprogrammers to perform their built in selftests. Note that 6942A/6943A selftests include I/O card testing as part of the procedure, not as separate tests as with a 6944A system.

All I/O cards which are recognized are listed on the screen along with their frame and slot address and a pass or fail message for status. Cards which pass the tests will show the message "passed", and those which fail will show the error code (−52 to −61) associated with the failure. There are two additional I/O card error codes which could occur. Error −50 means that there is no response from that slot in the Multiprogrammer. Error −51 means that the frame which contains that slot did not respond, thus all slots in that frame will return that error.

In certain cases an I/O card failure will cause the I/O card to return an invalid identification code. If this happens, the program will return the message "I/O card failure" where the card type (name) would have been displayed.

See the section of "More on Test_multi" at the rear of this Appendix.

The figure below illustrates some of the more common System__selft messages:

```
 6944A Multiprogrammer System:

********** SELFTEST OF FRAME(S) AT  29 **********
HP98633A Interface at Select Code 29
         Passed test.

HP6944A Multiprogrammer- Frame 0
  0      3  HP69751A 33kHz A/D Card          passed
  0      5  HP69720A D/A Converter           passed
  0     11  HP69735A Pulse Output Card       passed
  0     12  HP69775A Counter/Totalizer Card  passed
         Frame Passed Test.

********** SELFTEST OF FRAME(S) AT  30 **********

HP98633A Interface Selftest at Select Code  30
         Failed test : error -1
```

```
 6942A Multiprogrammer System:

********* SELFTEST OF FRAME(S) AT  2?  *********
 Failed Self Test.
6942A/43A Multiprogrammer-  Frame  0
823        3      HP69790B 4K Word Memory System    failed -56
823        8      HP69731B Digital Output           passed

********* SELFTEST OF FRAME(S) AT  3?5  *********
 Timeout on  825
```

Even though this program is part of the 14752A Verification procedure it may be used in other programs.

If "VERIFY" has not been loaded into the computer and you wish to perform this selftest, insert the 14752A Utilities disc into the disc drive and type LOADSUB ALL FROM "SELFT" and press EXECUTE. Now type CALL System__selft ("ZZZZ","YYYY") where "ZZZZ" is the name of the configuration file you are using and "YYYY" is the mass storage device specifier of the disc drive in which the configuration file disc is located.

If "VERIFY" has been loaded into the computer, then just type in CALL System__selft ("ZZZZ","YYYY") and press EXECUTE to perform this selftest. Note that "ZZZZ" and "YYYY" have the same meaning as above.

Normally you should verify the operation of your system one Multiprogrammer frame at a time. However, in your own application more than one Multiprogrammer may be involved. The System__selft utility is designed for this general use.

Once all selftests have been completed (pass or fail) the screen will display "ALL SELFTESTS COMPLETED". When you are done examining the results of the selftest, you may either press the "CONTINUE" softkey to continue with the verification or RESET the computer and try to fix any failures which have occurred.

The program then asks you to insert the 14752A Utility Programs disc and waits until you have done so and have pressed softkey the "CONTINUE" softkey. You have now reached the actual beginning of the functional test section.

Functional Testing

"VERIFY" calls up a version of the front panel similar to what you have already seen. A copy of a sample "TEST FRONT PANEL" is shown in Figure A-1.

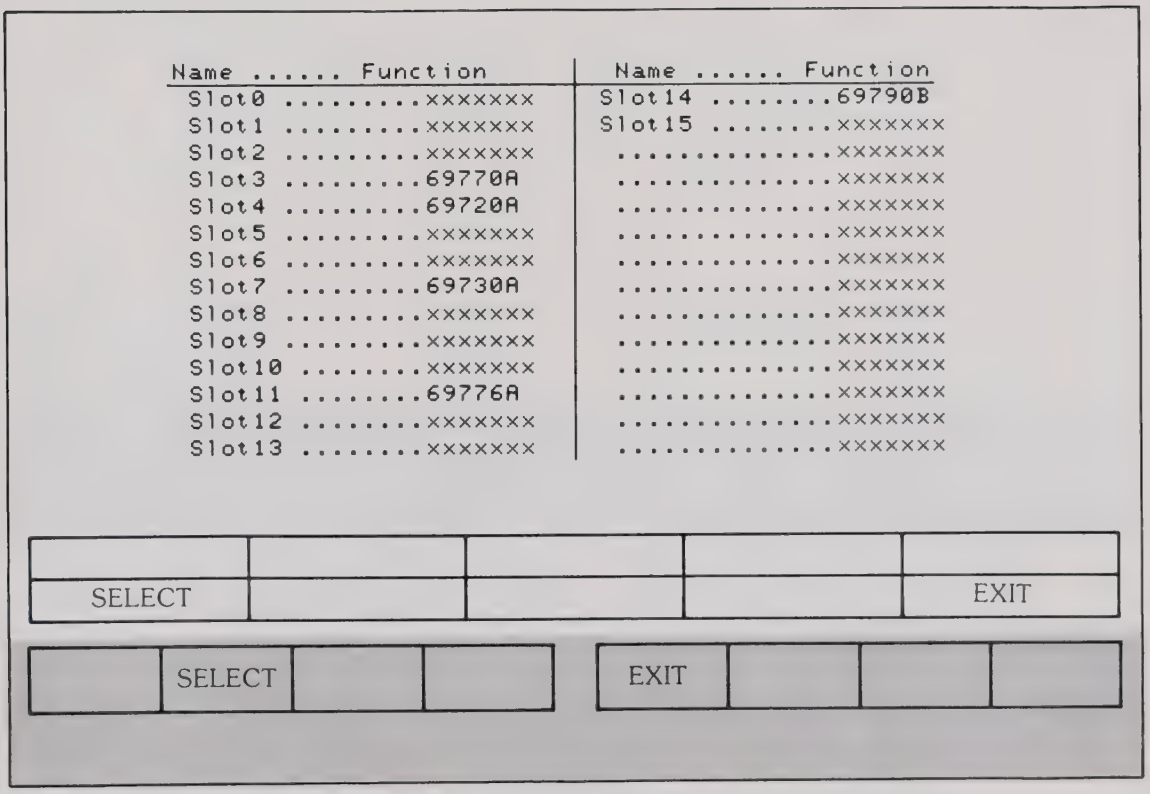


Figure A-1. "TEST FRONT PANEL"

Defined names will show what type of I/O card function they are associated with; undefined names will show "xxxxxxx" as a function. By using the cursor control keys move the cursor to the defined name you wish to test, then press softkey "Select" to enter the test program for that type of function. Selecting an undefined name, pressing any keyboard key, or any unlabeled softkey will cause the computer to emit a "beep" without any interference to the program in progress. If you press "EXIT" softkey, the program will stop. Selecting a defined name causes the program to advance to the "CARD IDENTIFICATION" screen where the selected function type is displayed and test set-up instructions may be requested.

A sample "CARD INDENTIFICATION" screen is shown in Figure A-2.

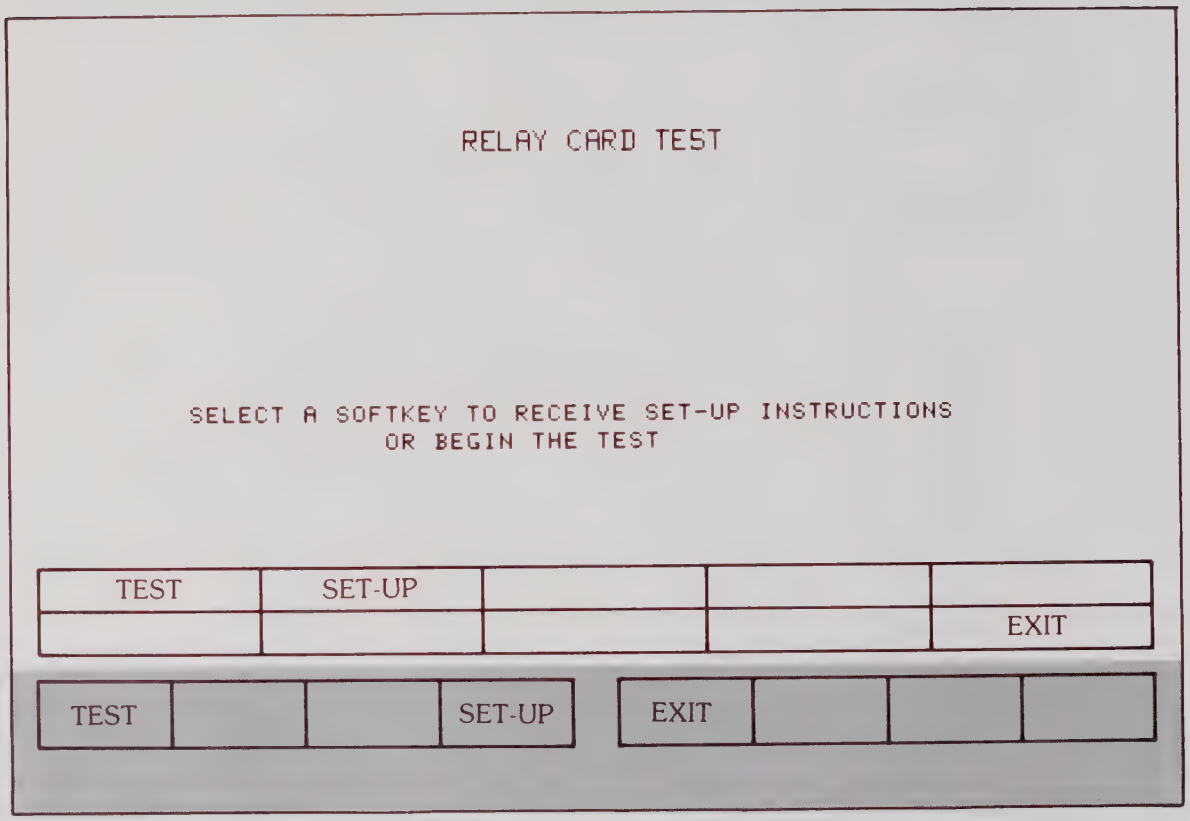


Figure A-2. "CARD IDENTIFICATION" screen for a 69730A Relay Output Card

Selecting softkey ("EXIT") returns you to the "TEST FRONT PANEL". Selecting "SET-UP" softkey (see Figure A-3) will print instructions concerning the test specifics and a description of the hardware and interconnections required to perform the test. When the setup has been completed or the instructions have been read, press softkey ("CONTINUE") to begin the test. Note that the "TEST" softkey in the "CARD IDENTIFICATION" screen causes the program to skip the set-up instructions and proceed directly to the functional test. It is recommended that if you are unfamiliar with these programs, that the "SET-UP" screen be reviewed before proceeding to the test.

Figure A-3 shows a sample "SET-UP" screen.

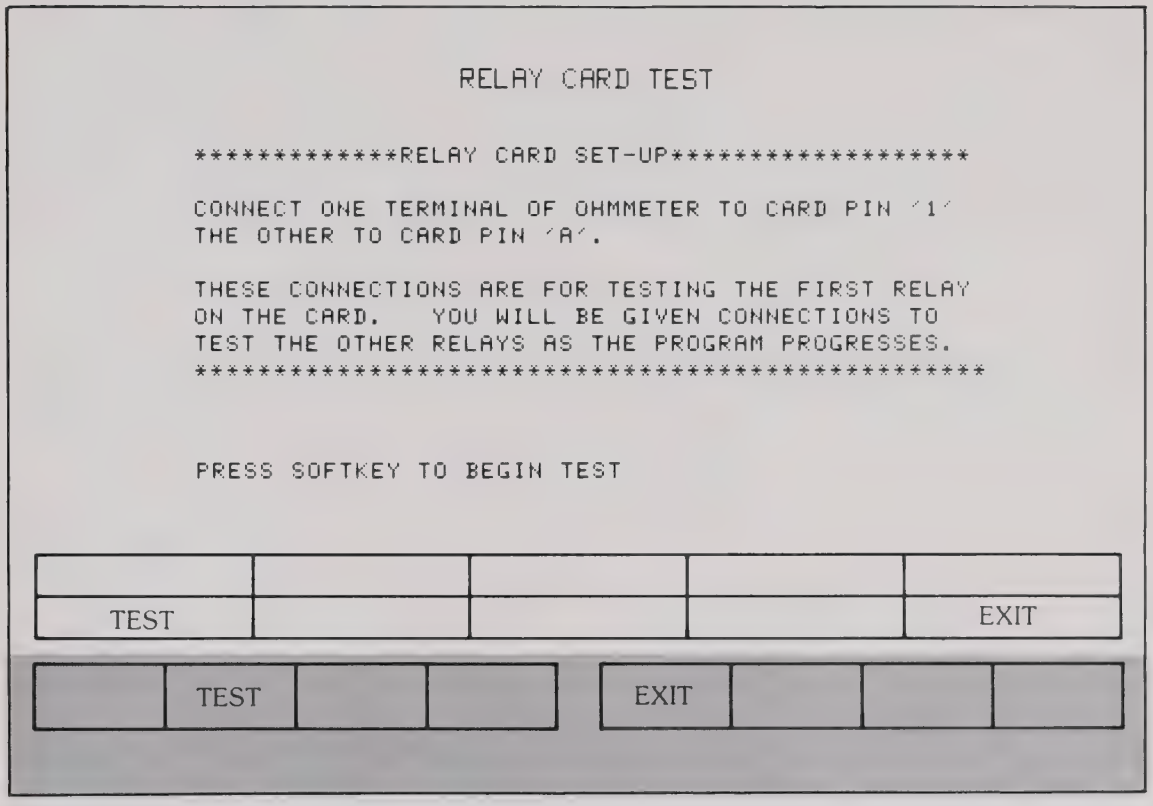

RELAY CARD TEST
**********RELAY CARD SET-UP**********
CONNECT ONE TERMINAL OF OHMMETER TO CARD PIN '1'
THE OTHER TO CARD PIN 'A'.
THESE CONNECTIONS ARE FOR TESTING THE FIRST RELAY
ON THE CARD. YOU WILL BE GIVEN CONNECTIONS TO
TEST THE OTHER RELAYS AS THE PROGRAM PROGRESSES.
****
PRESS SOFTKEY TO BEGIN TEST
TEST
EXIT
TEST
EXIT

Figure A-3. "SET-UP" screen for a 69730A Relay Output Card

The "TEST" screens for the I/O cards vary according to the function being exercised. In most cases, however, the topical information displayed during a test is similar to other functions. The "TEST" screen for the 69730A Relay Output card is shown in Figure A-4.

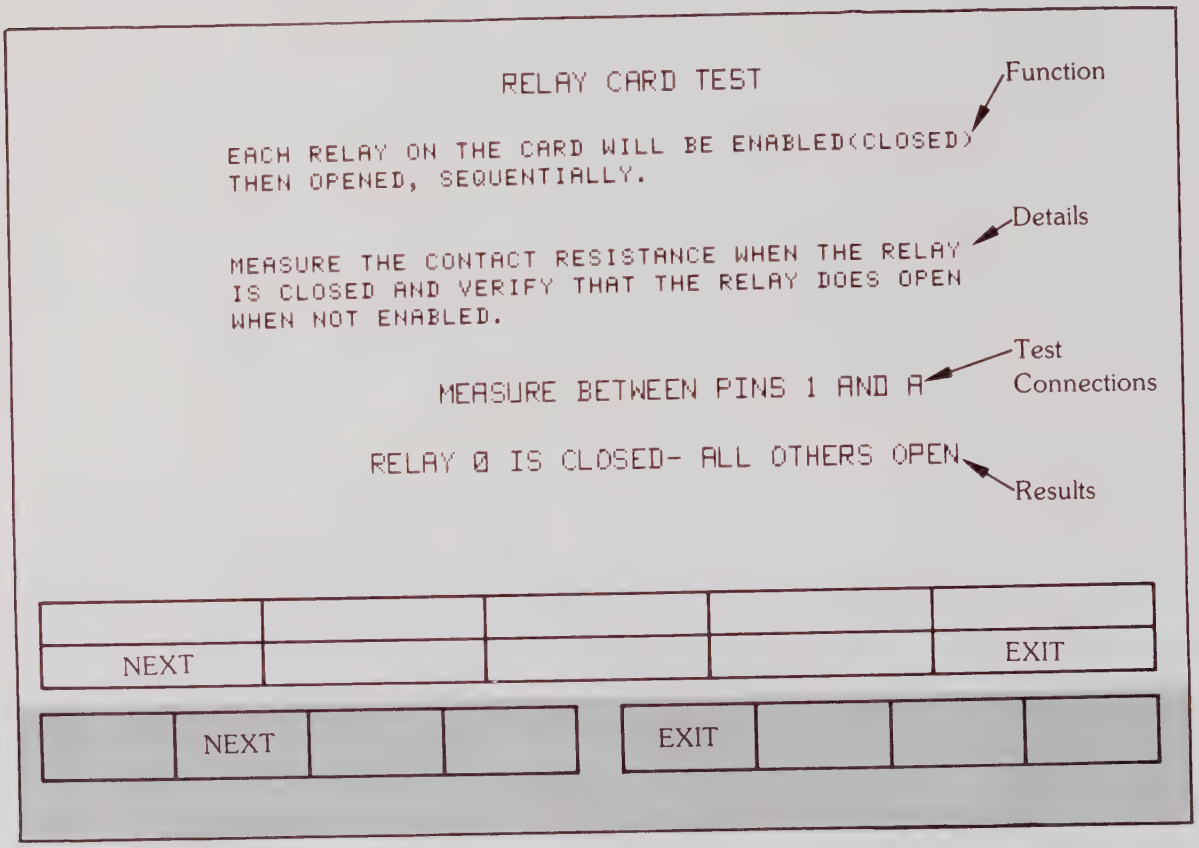


Figure A-4. "TEST" screen for a 69730A Relay Output Card

The screen is divided into several sections — function, details, test connections and test results. The "function" section is provided as a reminder of what type of I/O card is being tested. The "details" section gives information on what interaction (testing) is required by the operator at this point in the program. The "test connections" section gives I/O card edge connector pin locations to monitor (where applicable). Many of the I/O cards have only one set of connections to be made which suffice for performing the entire verification. These cards do not show set-up/test connections on the test screen, they are given in the SET-UP instructions. Cards which have test connections which changes as the test progresses (like the 69730A Relay Output Card) will show the connections for each portion of the test on the "TEST" screen. Selecting the "EXIT" softkey returns you to the "TEST FRONT PANEL".

The contents of the results field varies with the test in progress and is discussed in the "Individual Tests" section.

Note that all cables etc. must be removed from the edge connector of the I/O card before actually beginning the test. It is advisable to plug the I/O Card Edge Connector Test Assembly onto the I/O card's edge connector prior to selecting the functional test program from the "TEST FRONT PANEL".

Individual Tests

Gate-Flag Testing

Prior to beginning the interactive testing portion of the program a gate-flag test is performed on many of the cards.

A gate-flag test consists of cycling the selected I/O card (making it perform its function) and examining the status of several of its control signals to determine if it has actually completed operation. If the card does not complete operation; i.e., the gate-flag handshake is not completed, a message is printed stating that the use of the Input or Output 14752A commands is not recommended. The function will timeout if these commands are used. If no timeout has been specified for the function, then it will wait forever and hang up the computer.

A substitute sequence of commands to perform the same function without timing out is presented if there is one.

NOTES

Several of the I/O cards have on-board switches or jumpers which open the gate-flag handshake when external control of the I/O card is desired. If the program presents the message concerning open gate-flag, make sure that the handshake has not been disabled by a switch or jumper.

Several of the cards have switches which disable the output/input function of the card while still allowing the card to respond to the backplane trigger signal. The output of these cards is only enabled when an external trigger signal is received.

There is no way to test for this condition. Cards to which this condition applies will print a message such as "If there is no output from the card consult Appendix A" on the "TEST" screen. The write-up on the card in this Appendix will give the proper switch settings for the card to be tested or refer to the Operating and Service Manual for that I/O card.

Test Details

The I/O cards supported by the 14752A are divided into several categories according to their function. In the following pages the functional groups; Input, Output, Timing, and Memory, Interrupt and Scanning are discussed in terms of the I/O cards associated with that type of function and the tests performed on those cards.

Input Card Functions

In general, input card test programs perform a looping read of the card and display the value read on the computer CRT. Supported Input cards are:

1. 69771A Digital Input/Analog Comparator
2. 69770A Isolated Digital Input
3. 69751A 33 kHz A/D
4. 69759A 500 kHz A/D

69771A Digital Input/Analog Comparator

The digital input test performs a looping read of the card at the input edge connector (Pins A to T) and displays the result as a 16 bit binary number (0's and 1's) showing the logic level of the inputs. Note that for a standard card (no on-board logic inversion) the display will show all ones (with no connections). If the card has been set up for inverse logic, the display will be zeros. Touching a clip lead or wire connected to pin 36 to one of the input bits should invert the displayed value of that bit. Check the entire card by touching the wire to each bit in turn. The rightmost digit in the binary number is for Pin "A" and the leftmost digit is for "T".

NOTE

The digital input card test is preceded by a gate-flag test.

69770A Isolated Digital Input

The isolated digital input test program uses the same general procedure as the 69771A Digital Input function to test the card; i.e, a looping read with the results displayed in the same format. However, the isolated inputs require that an external source be applied to a bit before the bit can be driven Lo. Instructions are given in the program to accomplish this easily. Note, the program will display all 0's when no connections are made rather than all 1's. The rightmost digit in the binary number is for pin "A" and the leftmost digit is for "T".

NOTE

The isolated digital input card test is preceded by a gate-flag test.

69751A 33 kHz A/D Card

Verification. The 33 kHz A/D test program performs a looping read of the input terminals on the card edge. Check the card by applying input voltages to the card and comparing the readback to the actual input value. Each of the switch selectable range and the attenuator circuit can be checked by configuring the card with a different name and changing the lsb value in the menu to the appropriate value. Then perform verification on each of the "names". Select the "EXIT" softkey returns you to the CARD ID screen. From there the calibration (softkey "CAL") routine can be selected, testing be repeated (softkey "TEST"), or you may return to the TEST FRONT PANEL (softkey "EXIT"). PANEL (Softkey — "EXIT").

When configuring the card for testing, enter information for Multiprogrammer type, Interface Select Code, Frame, Slot, Full-scale range (also Range and LSB Value, if necessary) only. Do not change any other default values in the menus.

NOTE

A gate-flag check is performed prior to the A/D test. If gate-flag is open (switch S2-1 open) the test program cannot be run. If this is the case, a message stating that "GATE-FLAG IS OPEN" and "THIS TEST CANNOT BE PERFORMED" is displayed and the program returns to the "TEST FRONT PANEL". To test the card, turn off the Multiprogrammer, check the switch settings (S2-1 closed), and re-run "VERIFY".

Calibration. Available as part of the function test for the 69751A 33 kHz A/D card is a calibration routine which is accessed by selecting "CAL" softkey in the card identification screen. The calibration program uses a number of screen displays to explain the equipment set-up and user interaction required during the performance of the checks. Calibration steps include Input Amplifier Offset and Gain adjusts and A/D Offset and Gain adjustments. All programming of the A/D as required by the calibration process is handled by the program.

Supported A/D card ranges for calibration are listed in the 69751A Operating Manual, Section 5. Details on the calibration hardware are available in the 69751A Operating Manual and the beginning of this appendix.

Following the calibration procedure, the program returns to the Card ID screen so that verification can be performed if desired.

69759A 500 kHz A/D Card

Verification. The 69759A 500 kHz A/D test program performs a looping read of input terminals on the card edge. The display will be the voltage value present at the input of the A/D. Check the card by applying input voltages to the card and comparing the readback to the actual input value.

When configuring this card for testing, enter full-scale range in addition to the slot, frame, interface select code and Multiprogrammer type. This information is used for both the functional test and calibration sections of the verification program.

NOTE

There is no gate-flag test for this card.

Calibration. Available as part of the functional test for the 69759A 500 kHz A/D card is a calibration routine which is accessed by selecting softkey "CAL" in the card identification screen. The calibration program uses a number of screen displays to explain the equipment set-up and user interaction required during the performance of the checks. Calibration steps include Conversion Time Adjust, Input Amplifier Offset and Gain adjusts, A/D Offset and Gain adjustment and Attenuator Frequency Compensation Adjustments. All programming of the A/D as required by the calibration process is handled by the program.

Supported ranges for calibration are listed in the 69759A Operating Manual, Section 5. Details on the calibration hardware are available in the 69759A Operating Manual and the beginning of this appendix.

Following the calibration procedure, the program returns to the Card ID screen so that verification can be performed if desired.

Output Card Functions

There are two general types of output card tests. The first type of test (Digital Output card) sends several alternating data patterns to the card to test it's output capabilities. The patterns will repeat until you press the "EXIT" softkey.

The other type of test sequentially enables the outputs of the I/O card under test. For Relay and Resistance cards each channel (relay or resistor) will be enabled individually. Then all channels will be enabled, then disabled. The D/A function test will ramp the output from 0 to positive full scale, then to negative full scale.

Supported output cards are:

1. 69731B Digital Output
2. 69720A/721A Voltage and Current D/A
3. 69700A-706A Resistance Output
4. 69730A Relay Output
5. 69709A Power Supply Control

69731B Digital Output

The digital output test consists of alternately writing two data patterns "0101010101010101" and "1010101010101010" to the card at one second intervals. This will cause each output to toggle. Check the card by monitoring each bit with a logic probe, scope, or voltmeter to make sure that it toggles from logic Low to logic High. The display will show "EXPECTED OUTPUT PATTERN IS" and one of the above patterns. The rightmost digit in the binary number is for "A" and the leftmost digit is for "T".

NOTE

The digital output card test is preceded by a gate-flag test.

69700A-69706A Resistance Output

The resistance output test will enable each of the 12 resistors on the card sequentially, then enable all of them, then disable all of them. As the test progresses the screen will display the expected resistance value (based on the type of card and including the 10 ohm series output resistor). The test advances to the next output pattern when the user presses the softkey ("NEXT"). Measure each resistance value using a DMM or an Ohmmeter and compare the result with the expected output of the card (as displayed on the screen).

If you are testing a 69700A Custom Resistance card then you must enter the LSB of the card during the configuration process. Even though entry of the LSB of the card is allowed for this program, this program still requires that the card be loaded with resistors in a binary weighting pattern (1LSB-2LSBs-4LSBs-...).

NOTE

The resistance card test is preceded by a gate-flag test.

69730A Relay Output

The Relay card test is very similar to the Resistance card test. Each relay in succession is closed, then all relays are closed, and then all relays are opened. The user is given edge connector test points to monitor, and is instructed to make sure all relays do open and close, and that the contact resistance of closed relays is an acceptable value (contact resistance for a closed relay is .28 to .33 ohms). Measure the contact resistance with a DMM or Ohmmeter.

NOTE

There is no gate-flag test for this card.

69720A/721A Voltage/Current Digital to Analog Converter

Both the Voltage D/A and Current D/A card use the same program. The user is instructed to monitor the output of the card as the card is programmed to ramp from 0 to positive full scale, then be set to negative full scale. The display shows the expected output value of the card, in volts for a Voltage D/A and milliamperes for a Current D/A; pressing "NEXT" softkey will program the next output value. output value.

This is one of the cards that has an external trigger switch, which when enabled, does not allow the card's output to be programmed from the backplane. If there is no output during the test, check S1; S1-1 must be closed for the card to be able to accept backplane triggers.

NOTE

There is no gate-flag test for this card. However, the output of the card can be disabled until an external trigger is received. If there is no output from the card, check the setting of switch S1; S1-1 should be closed.

69709A Power Supply Control

There is no program included for this card. To test the card, use the 14752A Soft Front Panel to output various voltages and currents, and observe the operation of the power supply.

Timing Functions

The I/O cards associated with timing functions include both input and output cards. Output type cards will output a series of pulses (or a square wave) and display information concerning the period and number of pulses programmed. The timing input will display the value of the counter on the screen. The timing function cards include:

1. 69735A Pulse Train Output
2. 69736A Timer/Pacer Output
3. 69775A Counter/Totalizer

69735A Pulse Train Output

The test program for the Pulse Train card outputs a series of pulses from the positive square wave and pulse output pins, then outputs a similar pattern from the negative output pins. Each type of pulse lasts for 5 seconds, then the pattern is repeated so that the pulse and square wave outputs can be checked with positive and negative outputs programmed. The best procedure is to monitor the square wave output terminals (first positive then negative) with an oscilloscope or logic probe, then as the sequence repeats monitor the pulse output terminals.

NOTE

There is no gate-flag test for this card. However, the output of the card can be disabled until an external trigger is received. If there is no output from the card, check the setting of switch S1; S1-1 should be closed.

69736A Timer/Pacer Output

The Timer/Pacer test causes the card to output 3 different period waveforms. The three periods are 50uS, 50 mS and 1 S. Ten seconds of each type of pulse are output. The screen displays "Set up scope for waveform with period...", then waits for you to press the "CONTINUE" softkey to enable the output. While the waveform is being output, the screen displays the message "Output period is...". After 10 seconds of output the screen again displays "Set up the scope for waveform with period..." and the next value for the period. Monitor the output with a scope or Logic Probe.

NOTE

The Timer/Pacer test is preceded by a gate-flag test. However, the output of the card can be disabled until an external trigger is received. If there is no output from the card, check the setting of switch S1; S1-1 should be closed.

69775A Counter/Totalizer

The counter card test is simply a looping read of the count register. The card should be set up to enable both up and down count registers (switch banks S1 and S2 set as they were when shipped from the factory and documented in the Operating Manual for the card). With the switches set in this manner, the count register increments (decrements) on the edge of the input signal. Connect a clip lead to COM (pin 36 or r) and touch the other end to one of the counter inputs. Check the count to make sure that it can change in both the positive and negative directions. The screen will display "Present value of counter is...". Pressing the "EXIT" softkey will exit the test. Note that if the counter reaches either 0 or 65535, then the count may "Roll over", ie +1 = +65535 and 65535 +1 = 0. Touching the wire or clip lead to the input may cause the count to change by more than 1 since the card is very sensitive to this kind of "noisy" signal. This is normal operation.

NOTE

There is no gate-flag test for this card.

Memory, Interrupt and Scanning Functions

The Memory, Interrupt and Scanning cards close out the Multiprogrammer I/O card test programs.

69790B 4K Memory

The memory card test requires no interaction on your part. The test consists of several Write/Reads to check the integrity of the memory card address and data lines. The address test writes the address of each memory location into that location as data, then reads it back and compares it to the expected data. The Write/Read RAM test writes several data patterns to the memory card, then reads them back and compares them to the data pattern sent. If either of these tests fails, the test name is printed and the message "Memory Card Tests Failed" is displayed. If all tests are passed, the message "Memory Card Tests Passed" is displayed.

NOTE

There is no gate-flag test for this card.

69791A/69792A Memory System

This memory card test requires minimal interaction on your part. The test is in two main parts, memory size and RAM test.

NOTES

RAM tests are only performed on 16 k (16384) words of memory at a time for reasons of brevity and computer memory considerations.

To properly perform Memory System Verification the number of 69792A Memory Expansion card should be specified as "5" in the menu. This is the addition to the entries for Multiprogrammer Type, Interface, Frame, Slot in the menu.

Following the CARD IDENTIFICATION SCREEN, and prior to reaching the TEST screen, a memory search occurs to locate any memory elements responding to communication. Once the available memory has been determined (and displayed) you may select the 16 k of RAM locations to receive RAM testing. Selecting the "CHANGE" softkey allows selection of a 16 k block of RAM other than that at 0 to 16383. When the "CHANGE" softkey is selected, you are asked to enter a memory location for testing. Enter any memory address value, the program will determine which block of 16 k memory locations your selection resides in and display starting and ending memory locations for that block.

Pressing the "TEST" softkey will test whatever memory addresses are listed as "MEMORY TO BE TESTED" on the screen.

The RAM test includes an address line test which writes the address of each memory location into that location as data and then reads it back and compares them to the expected value. The Write/Read RAM test writes several data patterns to the memory card, then reads them back and compares it

to the data pattern sent. If either of these tests fail, the test name is printed and the message "Memory Card Tests Failed" is displayed. If all tests are passed, the message "Memory Card Tests Passed" is displayed. If the test fails then an appropriate message is displayed. The program then returns to the TEST screen so that other blocks of memory can be selected for RAM tests.

NOTE

There is no gate-flag test for this card.

69776A Interrupt

This test is slightly different in format from the other tests described so far. The test uses pin "B" as the reference pin from which the interrupt modes <, >, =, <> are compared. To change this, press the set-up key and select the "CHANGE" softkey which will allow you to use pins "B" to "S" as the reference pin. Once you have either selected another pin or decided to use "B" and pressed the "TEST" softkey, all necessary instructions are displayed on the screen. For each test, one or more messages will be displayed asking that a clip lead that is connected to COM be connected to, or disconnected from certain pins. If a test is failed, the test number and test name are displayed when tests are complete and a final Pass/Fail message is displayed.

NOTE

There is no gate-flag test for this card.

69750A (69752A, 69754A, 69755A) Scanning

This is a more complex test from the users point of view than any of the other tests. This test allows you to set up the parameters for scanning; i.e., period, start and stop channel, internal or external pacing, and then run several operational programs to check the scanner card system. There are two routines which are key to the testing of a scanning system: SCAN and STEP. The "STEP" routine is used to advance the scanner system through a sequential scan one channel at a time so that each channel can be checked for continuity. This routine is only used in the internal pacing mode. Selecting this test will enable the start channel, display which channel is enabled, and present the "NEXT" softkey to enable the next channel. In this way each channel can be checked.

The "SCAN" routine performs a sequential scan from the start to stop channels. If the pacing is done internally (on board pacer), the speed will be the programmed speed. If the pacer is set up for external, the channel advancing will occur whenever the external trigger line is touched to COM (Pin 36). Thus you can scan at any speed or advance one channel at a time as in the "STEP" program. The "EXIT" softkey is available to exit the test.

Any of the above scanning tests can be performed with or without the presence of a 69752A, 69754A, 69755A scanner cards. The approach to verification of the scanner system can be to check out the Scan Control card signals without a scanner card present or to check out scanner card channel integrity by performing verification on a Scan Control and scanner card.

NOTE

This card has switches which can open the gate-flag handshake. To test both internal and external triggers switch S1-2 and S1-3 should be closed and all other switches open.

More on Test_multi

The Test_multi utility performs verification tests on the control and communications paths in a Multiprogrammer system. There are several different checks which are performed by Test_multi, depending on the type of Multiprogrammer which is to be tested.

6942A/6943A System

For 6942A's or 6943A's, Test_multi causes the addressed Multiprogrammers (and any associated 6943A's) to initiate the built in selftest routines. These built-in firmware selftests perform verification on the Multiprogrammer Mainframe(s) and each of the I/O cards present in the system. The "Test_multi" software simply monitors the selftest as it processes and displays the result, it takes no active roll in the test (other than initiating it). The two main types of errors that can occur are due to mainframe and I/O card failures, respectively. If the Multiprogrammer passes the initial stages (mainframe tests) of the selftest then individual I/O card tests are performed.

A sample of the printed results of a "Test_multi" is shown below.

```
6942A Selftest at HP-IB 823
            Passed test.
Frame    Slot   Description                    Status
  0        3    69751A 33 kHz A/D Card         passed
  0        5    69720A D/A Converter           passed
  0        6    69731B Digital Output Card     passed
  0        7    69730A Relay Actuator Card     passed
  0       11    69735A Pulse Output Card       passed
  0       12    69775A Counter/Totalizer Card  passed
End of selftest.
```

6944A System

For 6944A systems, Test_multi performs two types of tests on the specified Multiprogrammer. The first is an interface test and the second is the tests of the I/O cards in the system. The test of the I/O card is actually a slot by slot repetition of one single I/O card test. The routine "Test_multi" actually performs these tests (there are no built-in "selftests" on a 6944A).

Like the test performed on a 6942A Multiprogrammer, the results are displayed in two parts: Results of the "Interface" part of the tests followed by the results of each of the I/O card tests performed.

A sample of the test results of a "Test__multi" on a HP 6944A system is shown below:

```
6944A at Select Code 29
                Passed test.
Frame     Slot    Description                    Status
  0         3     69751A 33 kHz A/D Card         passed
  0         5     69720A D/A Converter           passed
  0         6     69731B Digital Output Card     passed
  0         7     69730A Relay Actuator Card     passed
  0        11     69735A Pulse Output Card       passed
  0        12     69775A Counter/Totalizer Card  passed
End of selftest.
```

What About Errors?

There are several types of messages which could appear if errors occur during Test__multi.

If the first part of Test__multi fails (mainframe and/or interface tests). Then an error code from −1 to −24 will be displayed (see Appendix B of this manual for complete description of the Error Codes). Unless the error is due to an I/O Card, (Error −15) no further testing will be performed. If Error −15, or if no errors occur then individual I/O card test results will be displayed.

If, during I/O card tests, error codes −50 (no card in slot) or −51 (no response from Multiprogrammers) occur, then no message is printed for those errors. This is because the program cannot tell the difference between a totally non-functional I/O card and an empty slot, or between a non-functional or missing Multiprogrammers. If, upon conclusion of Test__multi, the screen displayed does not show Slot, Frame, Description and status information for one or more of the I/O cards in your system then you most likely have received errors −50 or −51.

The "Flag" returned by Test__multi has the number of errors that occurred during the test.

Since Test__multi does not stop or pause at completion - it is up to you to place a "PAUSE" statement in application program after Test__multi if you wish to review the test results.

Additional Test Routines for 6944A Systems

For a 6944A, system, the interface and slot test portions of Test__multi can be executed individually. The subprogram (CSUB) which performs interface tests is called "Test__ifc" and can be executed by typing:

```
CALL Test__ifc (Select__code,X)
and pressing EXECUTE or RETURN
```

and displaying the value of X returned.

NOTE

The 14752A LIBRARY - either the version for BASIC 2.1, BASIC 3.0 or BASIC 4.0 must already be present in the computer.

Test__ifc performs tests primarily aimed at checking the communications between the computer and the 6944A. Of primary interest is the interrupt system and the data path from 98633A Interface, through the cable, to the 6944A's Backplane Translater board. Errors which occur due to a failure during the Test__ifc part of Test__Multi have error codes in the range of −1 to −8 and are fully documented in Section 4 of the 6944A Operating and Service Manual.

The subprogram (CSUB) which performs I/O card tests is called "Test__slot" and can be executed by typing:

CALL Test__slot (Select__code,Slot,X)
and pressing EXECUTE

then displaying the value of X which is returned by Test__slot.

Test__slot first checks to see if the 6944A Mainframe, which contains the addressed slot is responding. If it is not, then an Error −51 is returned by Test__slot. Meaning "Multiprogrammer does not respond." Test__slot next checks to see if there is any response from the addressed slot by asking any I/O card in that slot to return an "Identification Word" (or Card ID). If there is no response from that slot then an error code −50 is returned by Test__slot meaning "NO CARD IN SLOT". Ths also occurs if there was a total failure of an I/O card located in that slot. Following the identification of an I/O card in a slot, Test__slot then proceeds to check the data communications and control paths between the I/O card and the 6944A backplane. Errors found during this portion of Test__slot are given the codes −52 to −61. See Section 4 of the 6944A Operating and Service Manual for full details on these errors.

NOTE

When executing any of these utilities from live keyboard mode the following restrictions apply:

- Since variables cannot be created by the user from the keyboard (variables can only be defined by the program), it is legal to use only parameters that already exist in the current context.
- Constants may be used in the pass parameter list.
- When calling a CSUB or SUB subprogram from the keyboard, the CALL keyword is not optional.

Either of these routines may also be called by simply inserting them like a basic subroutine.

Appendix B
Error Messages

14752A Error Messages

The following is a list of 14752A error messages. For a complete description of each error message refer to Chapter 4.

801 Unsupported Model Error
802 Soft Limit Exceeded
803 Timeout Error
804 Array Size Too Small
805 Reserved-5
806 Illegal Data Conversion Option
807 Illegal Limit Checking Option
808 Interrupting Device Not Found
809 Reserved-9
810 Illegal Vector Number
811 Names Not All At Same Address
812 Undefined Name Reference
813 Names COM Block Not Found
814 Illegal Model Id
815 Use of unINITialized Function
816 Illegal Device Type
817 14752A System Error
818 Illegal Data Value
819 Illegal Initial Function
820 Illegal Initial Range
821 Illegal Edge In Define
822 HP-IB Bus Timeout
823 Illegal Initial Mode Specified
824 Illegal Data Type Specified
825 Wrong Memory Mode
826 Illegal Error Number
827 Low Limit > High Limit
828 Start + Length Too Long
829 Reserved-29
830 Multiprogrammer Not On-Line
831 Illegal Initial Trigger
832 Low Limit Too High
833 High Limit Too Low
834 Illegal Frame Error
835 Illegal Slot Error
836 Interface Card Not Present
837 Specified Card Not In Slot
838 Illegal Name: Not a Scaler REAL
839 Illegal Initial Auto Zero
840 Illegal Initial Interrupt Mask
841 Missing Address Buffer Memory
842 Illegal Initial Start Bit
843 Illegal Bit Number
844 Illegal Number of Digits
845 Wrong Scan Mode
846 No Range Info For Data Conversion
847 Illegal Range Value
848 Illegal Select Code
849 Interface Must Be a 98633A
850 Configuration Error
851 Buffer Must Have a Front End
852 Buffer Has an Incorrect Front End
853 Buffer Has an Incorrect Scanner
854 No Digout Controller in Timer
855 Full Scale Range Mismatch
856 Transmission System Failure
857 I/O Card Test Failure
858 Array Size Mismatch
859 Too Many Parameters

6944A Verification Error Codes

The following is a list of 6944A verification error codes. These codes only appear when the "Test_multi (select code)" statement is executed. Since this test consists of two parts, (as explained in Appendix A) two different error codes can be returned as shown below. The brief description that accompanies each error code below does not appear on the screen. Chapter 4 of the 6944A/98633A Installation and Service Manual gives a complete description of each error code.

Multiprogrammer Interface Errors

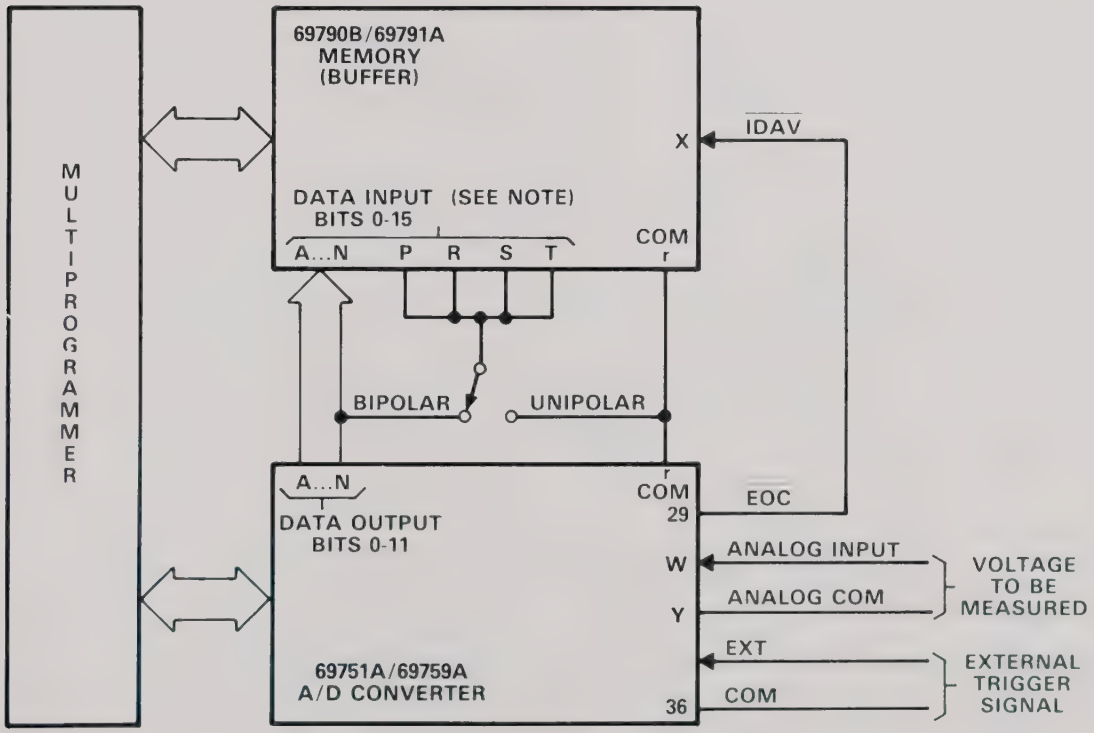
- **−1** No interface card found
- **−2** Interface not a 98633A
- **−3** 98633A defective
- **−4** 98633A defective
- **−5** 98633A defective
- **−6** 98633A, A4 board, or I/O card defective
- **−7** 98633A defective
- **−8** 98633A, A4 board, power supply, or cabling defective

I/O Card Errors

- **−52** Bad data readack
- **−53** ARM and EOP do not clear
- **−54** Data bus does not float high
- **−55** Bad I/O card identification
- **−56** 69790B Memory Card failed test
- **−57** 69790B Memory pattern sensitivity failure
- **−58** Data readback error
- **−59** Bad interrupt system
- **−60** ARM or EOP error
- **−61** No interrupt from I/O card

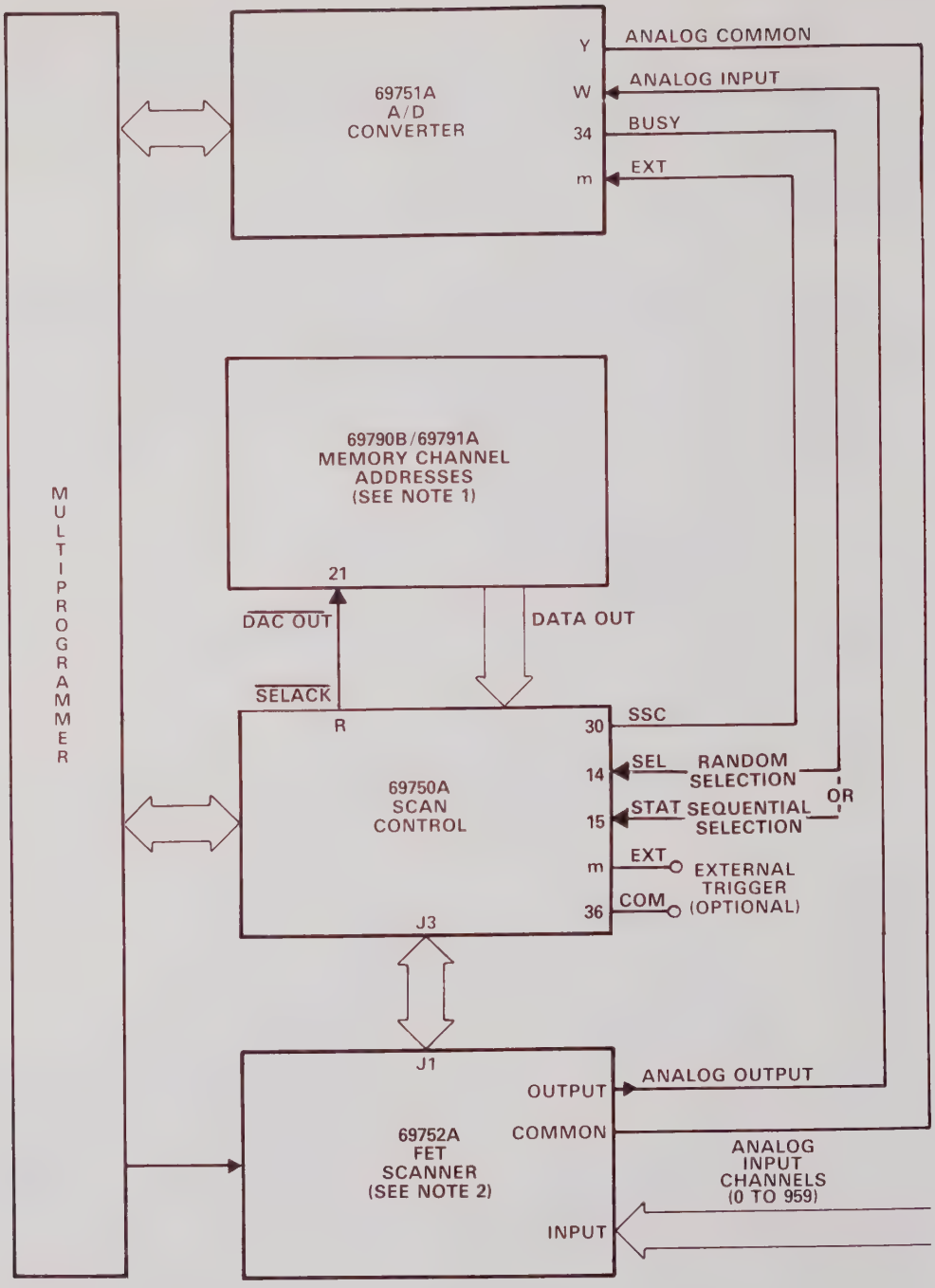
Appendix C
Multiple Card Interconnections

Throughout this manual there are configuration menus that allow more than one card to be used for one function. These multiple card functions are linked together by some common software. This common software assumes that the hardware is interconnected in a specific fashion. The wiring configurations for these functions are illustrated by the drawings in this Appendix. For detailed information and examples of the multiple card functions refer to Application Note 316-5 titled "Data Capture" (HP Part No. 5952-4109).



NOTE
THE FOUR MSB'S (PINS P,R,S,T) OF THE MEMORY DATA INPUT BITS MUST BE TIED TO THE MSB (PIN N) OF THE A/D CONVERTER DATA OUTPUT IF THE A/D IS OPERATING IN THE BIPOLAR RANGE; OR TO LOGIC 0 (PIN r) IF THE A/D CONVERTER IS SET TO THE UNIPOLAR RANGE. BIPOLAR IS THE DEFAULT RANGE ON THE A/D MENU PAGE 1.

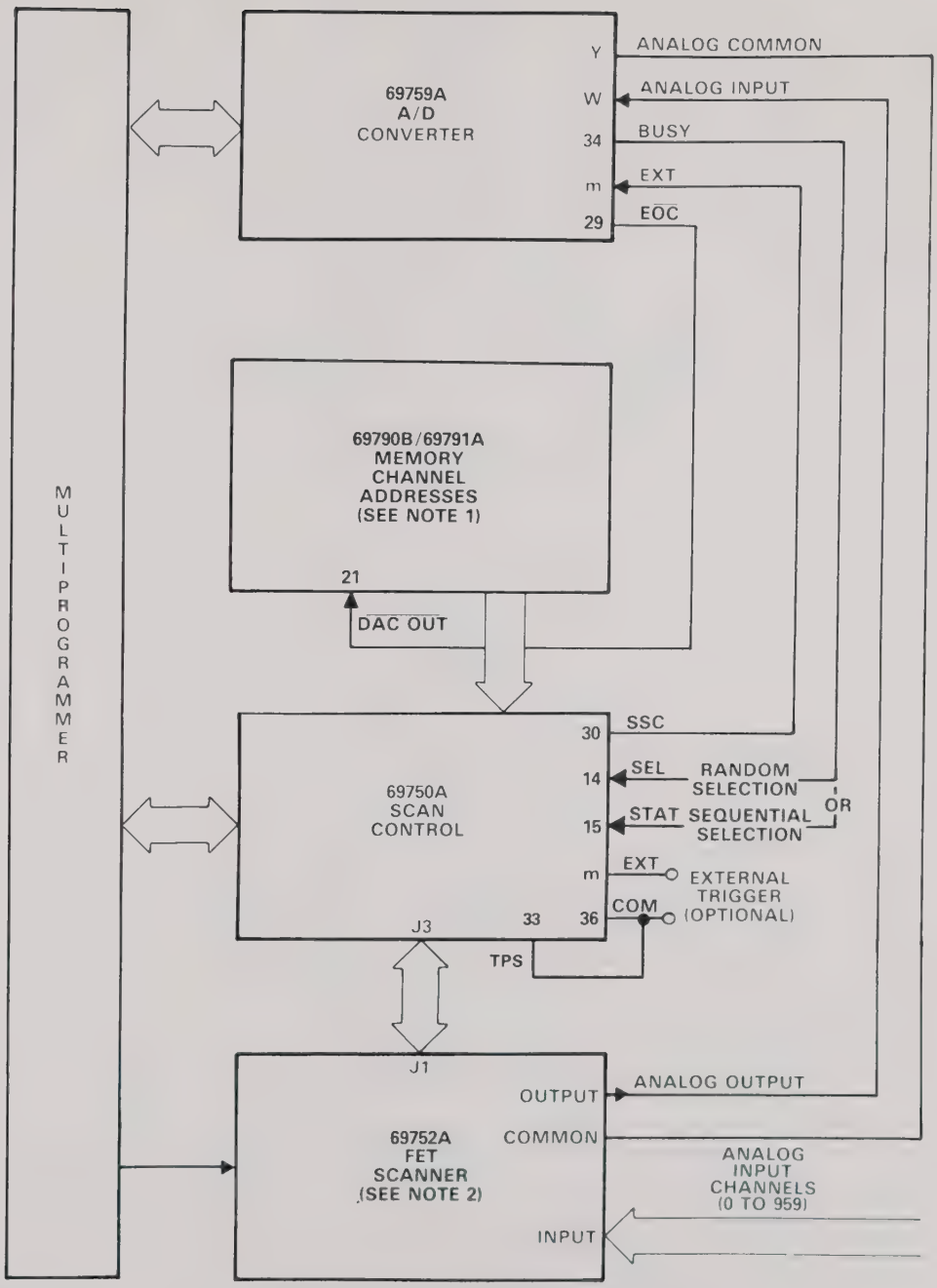
Figure C-1. Buffered A/D Function Interconnections



NOTES

1. USED ONLY IN RANDOM SCANNING MODE.
2. CAN ALSO BE A 69754A OR 69755A SCANNER CARD.

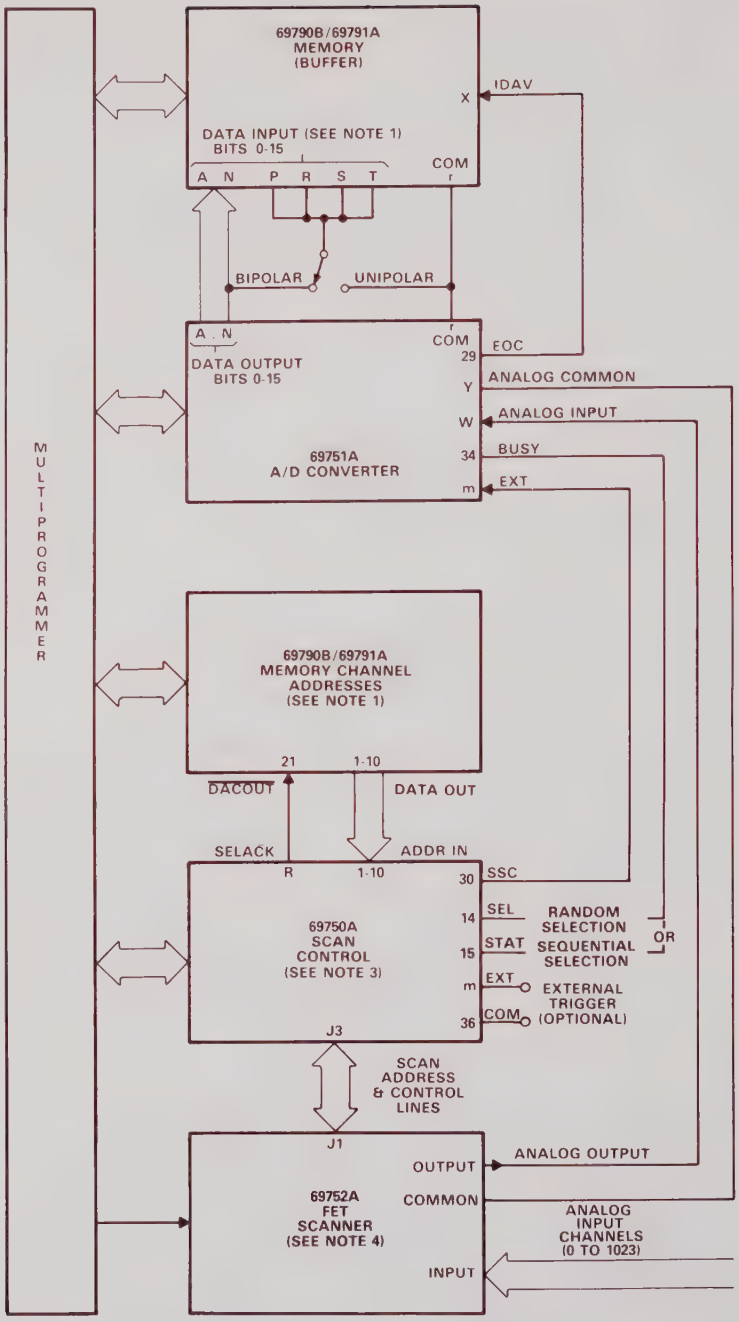
Figure C-2. Scanning A/D Function Card Interconnections (with a 69751A A/D Card)



NOTES

1. USED ONLY IN RANDOM SCANNING MODE.
2. CAN ALSO BE A 69754A OR 69755A SCANNER CARD.

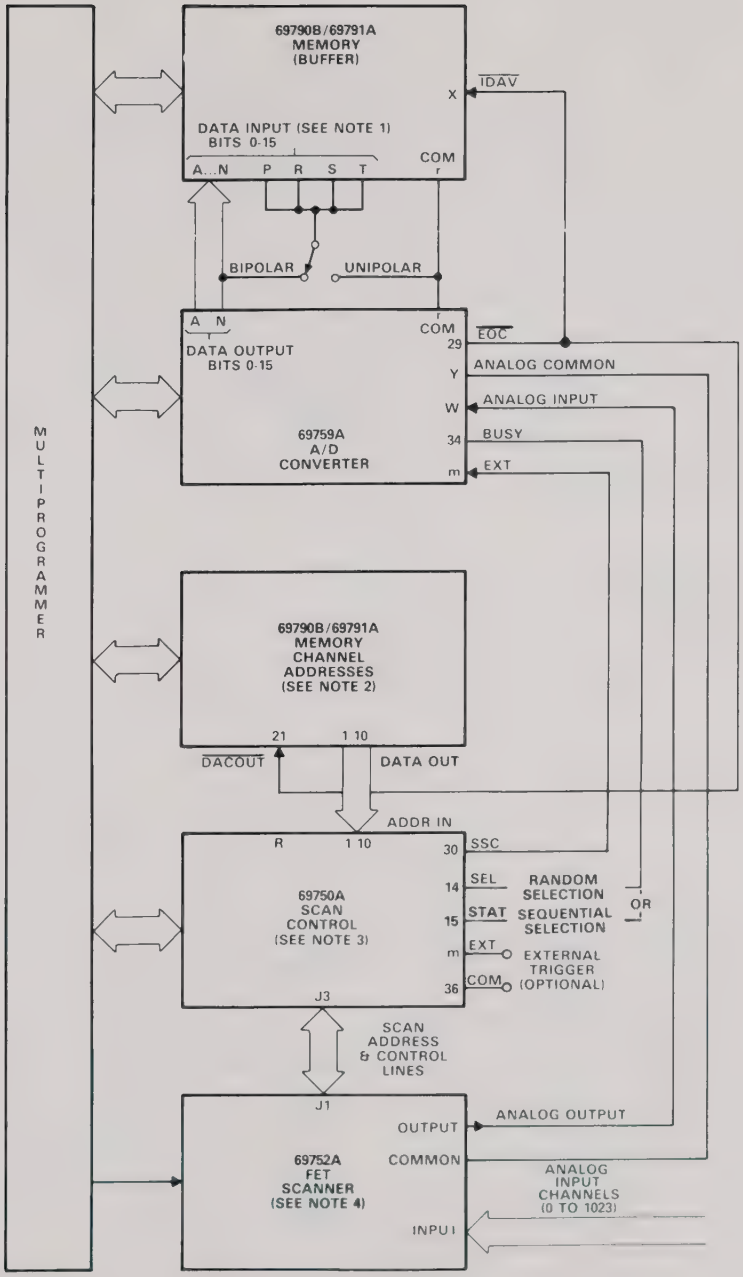
Figure C-3. Scanning A/D Function, Card Interconnections (with 69759A A/D Card)



NOTES

1. THE FOUR MSB'S (PINS P,R,S,T) OF THE MEMORY DATA INPUT BITS MUST BE TIED TO THE MSB (PIN N) OF THE A/D CONVERTER OUTPUT IF THE A/D IS OPERATING IN THE BIPOLAR RANGE; OR TO LOGIC 0 (PIN r) IF THE A/D IS SET TO THE UNIPOLAR RANGE. BIPOLAR IS THE DEFAULT RANGE ON THE A/D MENU PAGE 1.
2. USED ONLY IN RANDOM SCANNING MODE AND SELECTED AS ADDRESS MEMORY SLOT. THIS EXTERNAL CHANNEL MEMORY CAN BE THE SAME AS THE BUFFER MEMORY (ONLY WITH 69791A MEMORY).
3. THE PIN 14 (SEL) CONNECTION IS MADE FOR THE RANDOM SCANNING MODE. FOR SEQUENTIAL SCANNING, CONNECT PIN 34 (BUSY) ON THE 69751A A/D CONVERTER TO PIN 15 (STAT) ON THE 69750A SCAN CONTROL CARD AND REMOVE THE CONNECTION TO PIN 14 (SEC).
4. CAN ALSO BE A 69754A OR 69755A SCANNER CARD.

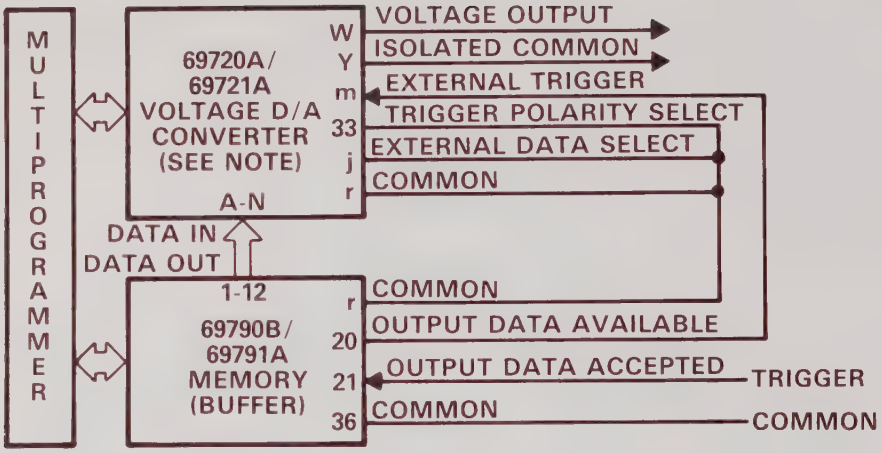
Figure C-4. Buffered Scanning A/D Function, Card Interconnections (with 69751A A/D Card)



NOTES

1. THE FOUR MSB'S (PINS P,R,S,T) OF THE MEMORY DATA INPUT BITS MUST BE TIED TO THE MSB (PIN N) OF THE A/D CONVERTER OUTPUT IF THE A/D IS OPERATING IN THE BIPOLAR RANGE, OR TO LOGIC 0 (PIN *r*) IF THE A/D IS SET TO THE UNIPOLAR RANGE. BIPOLAR IS THE DEFAULT RANGE ON THE A/D MENU PAGE 1.
2. USED ONLY IN RANDOM SCANNING MODE AND SELECTED AS ADDRESS MEMORY SLOT. THIS EXTERNAL CHANNEL MEMORY CAN BE THE SAME AS THE BUFFER MEMORY (ONLY WITH 69791A MEMORY).
3. THE PIN 14 (SEL) CONNECTION IS MADE FOR THE RANDOM SCANNING MODE FOR SEQUENTIAL SCANNING, CONNECT PIN 34 (BUSY) ON THE 69751A A/D CONVERTER TO PIN 15 (STAT) ON THE 69750A SCAN CONTROL CARD AND REMOVE THE CONNECTION TO PIN 14 (SEL).
4. CAN ALSO BE A 69754A OR 69755A SCANNER CARD.

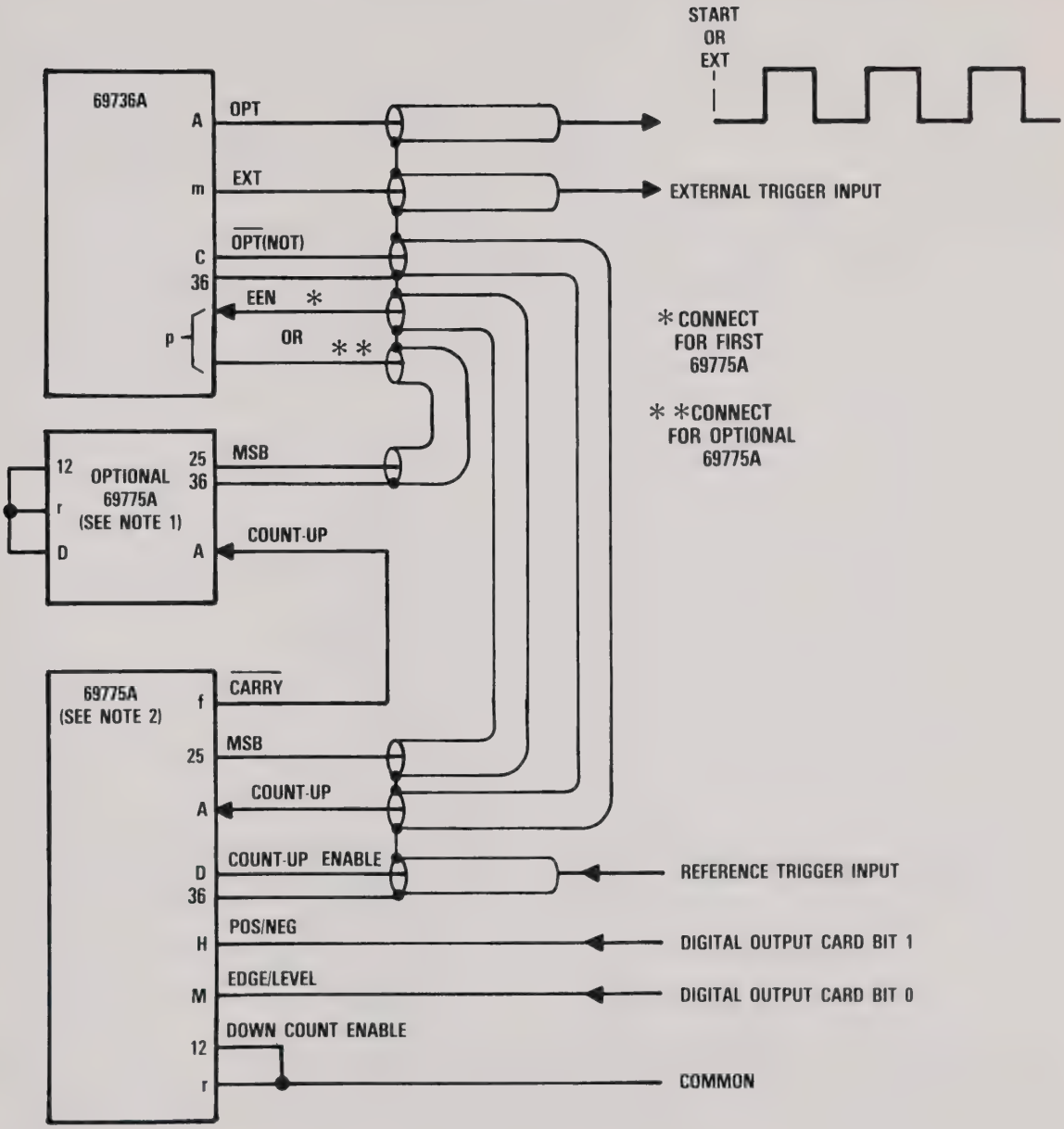
Figure C-5. Buffered Scanning A/D Function, Card Interconnections (with 69759A A/D Card)



NOTE

FOR A 69721A CURRENT D/A CONVERTER, THE OUTPUT CURRENT IS TAKEN AT PIN 19.

Figure C-6. Buffered D/A Function Interconnections



NOTES:

1 Switches on the optional HP 69775A card must be set as follows: S2-E open; S2-A,B,C,D,F closed.
2 When using an optional Digital Output card, open all switches except S2-F on this 69775A card. Otherwise, for Burst mode operation close only S2-D and S2-F; for Delay mode operation open S2-D, close S2-F

Figure C-7. Timer Function Interconnections

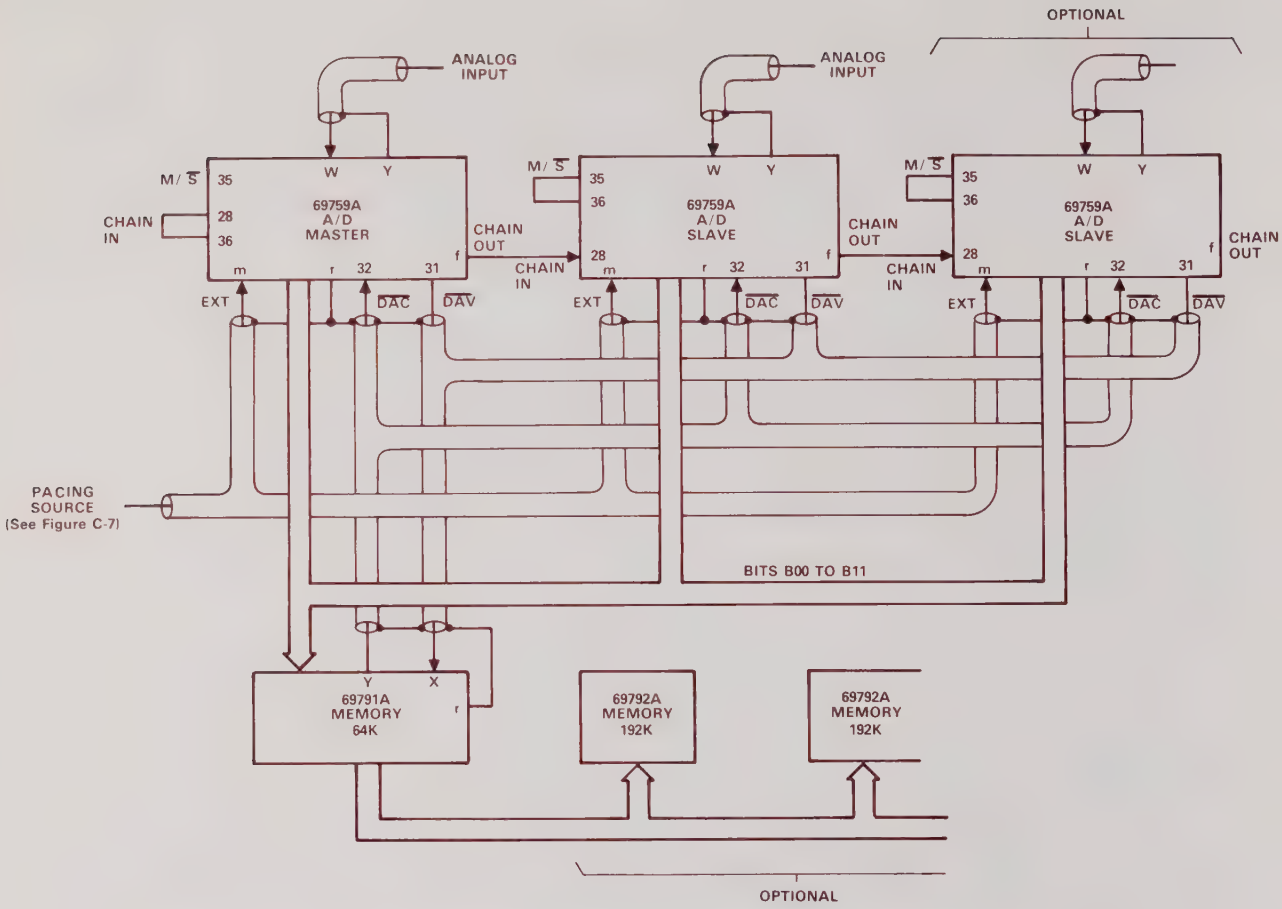


Figure C-8. Multiplexed Buffered A/D Function Interconnections

SALES & SUPPORT OFFICES

Arranged alphabetically by country

Product Line Sales/Support Key

Key Product Line
- **A** Analytical
- **CM** Components
- **C** Computer Systems Sales only
- **CH** Computer Systems Hardware Sales and Services
- **CS** Computer Systems Software Sales and Services
- **E** Electronic Instruments & Measurement Systems
- **M** Medical Products
- **MP** Medical Products Primary SRO
- **MS** Medical Products Secondary SRO
- **P** Personal Computation Products
- **·** Sales only for specific product line
- **··** Support only for specific product line

IMPORTANT: These symbols designate general product line capability. They do not insure sales or support availability for all products within a line, at all locations. Contact your local sales office for information regarding locations where HP support is available for specific products.

HP distributors are printed in italics.

HEADQUARTERS OFFICES

If there is no sales office listed for your area, contact one of these headquarters offices.

NORTH/CENTRAL AFRICA

Hewlett-Packard S.A.
7, Rue du Bois-du-Lan
CH-1217 **MEYRIN** 2, Switzerland
Tel: (022) 83 12 12
Telex: 27835 hpse
Cable: HEWPACKSA Geneve

ASIA

Hewlett-Packard Asia Ltd.
6th Floor, Sun Hung Kai Centre
30 Harbour Rd
G.P.O. Box 795
HONG KONG
Tel: 5-832 3211
After Jan. 1, 1984
47th Floor, China Resources Bldg.
26 Harbour Rd., Wanchai
HONG KONG
Telex: 66678 HEWPA HX
Cable: HEWPACK HONG KONG

CANADA

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
MISSISSAUGA, Ontario L4V 1M8
Tel: (416) 678-9430
Telex: 610-492-4246

EASTERN EUROPE

Hewlett-Packard Ges.m.b.h
Lieblgasse 1
P.O.Box 72
A-1222 **VIENNA**, Austria
Tel: (222) 2365110
Telex: 1 3 4425 HEPA A

NORTHERN EUROPE

Hewlett-Packard S.A
Uilenstede 475
P.O.Box 999
NL-1180 AZ **AMSTELVEEN**
The Netherlands
Tel: 20 437771

SOUTH EAST EUROPE

Hewlett-Packard S A
7, Rue du Bois-du-Lan
CH-1217 **MEYRIN** 2, Switzerland
Tel (022) 83 12 12
Telex. 27835 hpse
Cable. HEWPACKSA Geneve

OTHER EUROPE

Hewlett-Packard S.A.
P.O. Box
150, Rte du Nant-D'Avril
CH-1217 **MEYRIN** 2, Switzerland
Tel: (022) 83 8111
Telex: 22486 hpsa
Cable: HEWPACKSA Geneve

MEDITERRANEAN AND MIDDLE EAST

Hewlett-Packard S.A.
Mediterranean and Middle East Operations
Atrina Centre
32 Kifissias Ave.
Paradissos-Amarousion, **ATHENS**
Greece
Tel: 682 88 11
Telex: 21-6588 HPAT GR
Cable: HEWPACKSA Athens

EASTERN USA

Hewlett-Packard Co.
4 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 258-2000

MIDWESTERN USA

Hewlett-Packard Co
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800

SOUTHERN USA

Hewlett-Packard Co
2000 South Park Place
P.O Box 105005
ATLANTA, GA 30348
Tel. (404) 955-1500

WESTERN USA

Hewlett-Packard Co
3939 Lankershim Blvd
P.O. Box 3919
LOS ANGELES, CA 91604
Tel: (213) 506-3700

OTHER INTERNATIONAL AREAS

Hewlett-Packard Co
Intercontinental Headquarters
3495 Deer Creek Road
PALO ALTO, CA 94304
Tel: (415) 857-1501
Telex 034-8300
Cable HEWPACK

ANGOLA

Telectra
Empresa Técnica de Equipamentos
R. Barbosa Rodrigues, 41-I DT.
Caixa Postal 6487
LUANDA
Tel: 35515,35516
E,P

ARGENTINA

Hewlett-Packard Argentina S.A.
Avenida Santa Fe 2035
Martinez 1640 **BUENOS AIRES**
Tel: 798-5735, 792-1293
Telex: 17595 BIONAR
Cable: HEWPACKARG
A,E,CH,CS,P

Biotron S.A.C.I.M. e I.
Av Paseo Colon 221, Piso 9
*1399 **BUENOS AIRES***
Tel: 30-4846, 30-1851
Telex: 17595 BIONAR
M

AUSTRALIA

Adelaide, South Australia Office
Hewlett-Packard Australia Ltd
153 Greenhill Road
PARKSIDE, S.A. 5063
Tel: 272-5911
Telex: 82536
Cable: HEWPARD Adelaide
A*,CH,CM,,E,MS,P

Brisbane, Queensland Office
Hewlett-Packard Australia Ltd.
10 Payne Road
THE GAP, Queensland 4061
Tel: 30-4133
Telex: 42133
Cable: HEWPARD Brisbane
A,CH,CM,E,M,P

Canberra, Australia Capital Territory Office
Hewlett-Packard Australia Ltd
121 Wollongong Street
FYSHWICK, A.C.T. 2609
Tel: 80 4244
Telex: 62650
Cable: HEWPARD Canberra
CH,CM,E,P

Melbourne, Victoria Office
Hewlett-Packard Australia Ltd
31-41 Joseph Street
BLACKBURN, Victoria 3130
Tel: 895-2895
Telex. 31-024
Cable: HEWPARD Melbourne
A,CH,CM,CS,E,MS,P

Perth, Western Australia Office
Hewlett-Packard Australia Ltd
261 Stirling Highway
CLAREMONT, W.A 6010
Tel 383-2188
Telex 93859
Cable HEWPARD Perth
A,CH,CM,E,MS,P

Sydney, New South Wales Office
Hewlett-Packard Australia Ltd.
17-23 Talavera Road
P.O. Box 308
NORTH RYDE, N.S.W. 2113
Tel: 887-1611
Telex: 21561
Cable: HEWPARD Sydney
A,CH,CM,CS,E,MS,P

AUSTRIA

Hewlett-Packard Ges.m.b.h
Grottenhofstrasse 94
A-8052 **GRAZ**
Tel: (0316) 291 5 66
Telex: 32375
CH,E

Hewlett-Packard Ges.m.b.h
Lieblgasse 1
P.O. Box 72
A-1222 **VIENNA**
Tel: (0222) 23 65 11-0
Telex: 134425 HEPA A
A,CH,CM,CS,E,MS,P

BAHRAIN

Green Salon
P.O. Box 557
Manama
BAHRAIN
Tel: 255503-255950
Telex: 84419
P

Wael Pharmacy
P.O. Box 648
BAHRAIN
Tel: 256123
Telex: 8550 WAEL BN
E,C,M

BELGIUM

Hewlett-Packard Belgium S.A./N.V
Blvd de la Woluwe, 100
Woluwedal
B-1200 **BRUSSELS**
Tel: (02) 762-32-00
Telex: 23-494 paloben bru
A,CH,CM,CS,E,MP,P

BRAZIL

Hewlett-Packard do Brasil I.e.C. Ltda
Alameda Rio Negro, 750
Alphaville
06400 **BARUERI SP**
Tel: (011) 421 1311
Telex: (011) 33872 HPBR-BR
Cable HEWPACK Sao Paulo
A,CH,CM,CS,E,M,P

Hewlett-Packard do Brasil I e C. Ltda
Avenida Epitacio Pessoa, 4664
22471 **RIO DE JANEIRO**-RJ
Tel (021) 286 0237
Telex. 021-21905 HPBR-BR
Cable HEWPACK Rio de Janeiro
A,CH,CM,E,MS,P*

ANAMED I.C.E.I. Ltda
Rua Bage, 103
*04012 **SAO PAULO***
Tel (011) 570-5726
Telex 021-21905 HPBR BR
M

SALES & SUPPORT OFFICES

Arranged alphabetically by country

CANADA

Alberta
Hewlett-Packard (Canada) Ltd.
3030 3rd Avenue N.E.
CALGARY, Alberta T2A 6T7
Tel: (403) 235-3100
A,CH,CM,E*,MS,P*

Hewlett-Packard (Canada) Ltd.
11120A-178th Street
EDMONTON, Alberta T5S 1P2
Tel: (403) 486-6666
A,CH,CM,CS,E,MS,P

British Columbia
Hewlett-Packard (Canada) Ltd
10691 Shellbridge Way
RICHMOND,
British Columbia V6X 2W7
Tel: (604) 270-2277
Telex: 610-922-5059
A,CH,CM,CS,E*,MS,P*

Manitoba
Hewlett-Packard (Canada) Ltd.
380-550 Century Street
WINNIPEG, Manitoba R3H 0Y1
Tel: (204) 786-6701
A,CH,CM,E,MS,P*

Nova Scotia
Hewlett-Packard (Canada) Ltd
P.O. Box 931
900 Windmill Road
DARTMOUTH, Nova Scotia B2Y 3Z6
Tel: (902) 469-7820
CH,CM,CS,E*,MS,P*

Ontario
Hewlett-Packard (Canada) Ltd
3325 N. Service Rd., Unit 6
BURLINGTON, Ontario P3A 2A3
Tel: (416) 335-8644
CS,M*

Hewlett-Packard (Canada) Ltd.
552 Newbold Street
LONDON, Ontario N6E 2S5
Tel: (519) 686-9181
A,CH,CM,E*,MS,P*

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
MISSISSAUGA, Ontario L4V 1M8
Tel: (416) 678-9430
A,CH,CM,CS,E,MP,P

Hewlett-Packard (Canada) Ltd
2670 Queensview Dr.
OTTAWA, Ontario K2B 8K1
Tel: (613) 820-6483
A,CH,CM,CS,E*,MS,P*

Hewlett-Packard (Canada) Ltd.
220 Yorkland Blvd., Unit #11
WILLOWDALE, Ontario M2J 1R5
Tel: (416) 499-9333
CH

Quebec
Hewlett-Packard (Canada) Ltd.
17500 South Service Road
Trans-Canada Highway
KIRKLAND, Quebec H9J 2M5
Tel: (514) 697-4232
A,CH,CM,CS,E,MP,P*

Hewlett-Packard (Canada) Ltd.
Les Galeries du Vallon
2323 Du Versont Nord
STE. FOY, Quebec G1N 4C2
Tel: (418) 687-4570
CH

CHILE

Jorge Calcagni y Cia. Ltda.
Av. Italia 634 Santiago
Casilla 16475
***SANTIAGO** 9*
Tel: 222-0222
Telex. Public Booth 440001
A,CM,E,M

Olympia (Chile) Ltda.
Av. Rodrigo de Araya 1045
Casilla 256-V
***SANTIAGO** 21*
Tel. (02) 22 55 044
Telex: 240-565 OLYMP CL
Cable Olympiachile Santiagochile
CH,CS,P

CHINA, People's Republic of

China Hewlett-Packard Rep. Office
P.O. Box 418
1A Lane 2, Luchang St.
Beiwei Rd., Xuanwu District
BEIJING
Tel: 33-1947, 33-7426
Telex: 22601 CTSHP CN
Cable. 1920
A,CH,CM,CS,E,P

COLOMBIA

Instrumentación
H. A. Langebaek & Kier S.A.
Carrera 4A No. 52A-26
Apartado Aereo 6287
***BOGOTA** 1, D.E.*
Tel: 212-1466
Telex: 44400 INST CO
Cable: AARIS Bogota
CM,E,M

Casa Humboldt Ltda.
Carrera 14, No 98-60
Apartado Aereo 51283
***BOGOTA** 1, D.E.*
Tel: 256-1686
Telex: 45403 CCAL CO.
A

COSTA RICA

Cientifica Costarricense S.A.
Avenida 2, Calle 5
San Pedro de Montes de Oca
Apartado 10159
SAN JOSE
Tel: 24-38-20, 24-08-19
Telex: 2367 GALGUR CR
CM,E,M

CYPRUS

Telerexa Ltd.
P.O. Box 4809
14C Stassinos Avenue
NICOSIA
Tel: 62698
Telex: 2894 LEVIDO CY
E,M,P

DENMARK

Hewlett-Packard A/S
Datavej 52
DK-3460 **BIRKEROD**
Tel: (02) 81-66-40
Telex: 37409 hpas dk
A,CH,CM,CS,E,MS,P

Hewlett-Packard A/S
Rolighedsvej 32
DK-8240 **RISSKOV**, Aarhus
Tel: (06) 17-60-00
Telex: 37409 hpas dk
CH,E

DOMINICAN REPUBLIC

Microprog S.A.
Juan Tomás Mejía y Cotes No. 60
Arroyo Hondo
SANTO DOMINGO
Tel: 565-6268
Telex: 4510 ARENTA DR (RCA) P

ECUADOR

CYEDE Cia. Ltda.
Avenida Eloy Alfaro 1749
Casilla 6423 CCI
QUITO
Tel: 450-975, 243-052
Telex: 2548 CYEDE ED
CM,E,P

Hospitalar S.A.
Robles 625
Casilla 3590
QUITO
Tel: 545-250, 545-122
Telex: 2485 HOSPTL ED
Cable: HOSPITALAR-Quito
M

EGYPT

International Engineering Associates
24 Hussein Hegazi Street
Kasr-el-Aini
CAIRO
Tel: 23829, 21641
Telex: IEA UN 93830
CH,CS,E,M

EGYPOR
P.O.Box 2558
42 El Zahraa Street
***CAIRO**, Egypt*
Tel: 65 00 21
Telex. 93 337
P

EL SALVADOR

IPESA de El Salvador S.A.
29 Avenida Norte 1216
SAN SALVADOR
Tel: 26-6858, 26-6868
Telex: 20539 IPESASAL
A,CH,CM,CS,E,P

FINLAND

Hewlett-Packard Oy
Revontulentie 7
PL 24
SF-02101 **ESPOO** 10
Tel: (90) 4550211
Telex: 121563 hewpa sf
CH,CM,CS,P

Hewlett-Packard Oy
(Olarinluoma 7)
PL 24
02101 **ESPOO** 10
Tel: (90) 4521022
A,E,MS

Hewlett-Packard Oy
Aatoksenkatv 10-C
SF-40720-72 **JYVASKYLA**
Tel: (941) 216318
CH

Hewlett-Packard Oy
Kainvuntie 1-C
SF-90140-14 **OULU**
Tel: (981) 338785
CH

FRANCE

Hewlett-Packard France
Z.I. Mercure B
Rue Berthelot
F-13763 Les Milles Cedex
AIX-EN-PROVENCE
Tel: 16 (42) 59-41-02
Telex: 410770F
A,CH,E,MS,P*

Hewlett-Packard France
64, rue Marchand Saillant
F-61000 **ALENCON**
Tel: 16 (33) 29 04 42

Hewlett-Packard France
Boite Postale 503
F-25026 **BESANCON**
28 rue de la Republique
F-25000 **BESANCON**
Tel: 16 (81) 83-16-22
CH,M

Hewlett-Packard France
13, Place Napoleon III
F-29000 **BREST**
Tel: 16 (98) 03-38-35

Hewlett-Packard France
Chemin des Mouilles
Boite Postale 162
F-69130 **ECULLY** Cedex (Lyon)
Tel: 16 (78) 833-81-25
Telex: 310617F
A,CH,CS,E,MP

Hewlett-Packard France
Tour Lorraine
Boulevard de France
F-91035 **EVRY** Cedex
Tel: 16 6 077-96-60
Telex: 692315F
E

Hewlett-Packard France
Parc d'Activite du Bois Briard
Ave. du Lac
F-91040 **EVRY** Cedex
Tel: 16 6 077-8383
Telex: 692315F
E

Hewlett-Packard France
5, avenue Raymond Chanas
F-38320 **EYBENS** (Grenoble)
Tel: 16 (76) 25-81-41
Telex: 980124 HP GRENOB EYBE
CH

Hewlett-Packard France
Centre d'Affaire Paris-Nord
Bâtiment Ampère 5 étage
Rue de la Commune de Paris
Boite Postale 300
F-93153 **LE BLANC MESNIL**
Tel: 16 (1) 865-44-52
Telex: 211032F
CH,CS,E,MS

Hewlett-Packard France
Parc d'Activités Cadera
Quartier Jean Mermoz
Avenue du Président JF Kennedy
F-33700 **MERIGNAC** (Bordeaux)
Tel: 16 (56) 34-00-84
Telex: 550105F
CH,E,MS

Hewlett-Packard France
Immueble "Les 3 B"
Nouveau Chemin de la Garde
ZAC de Bois Briand
F-44085 **NANTES** Cedex
Tel: 16 (40) 50-32-22
CH**

SALES & SUPPORT OFFICES

Arranged alphabetically by country

FRANCE (Cont'd)

Hewlett-Packard France
125, rue du Faubourg Bannier
F-45000 **ORLEANS**
Tel: 16 (38) 68 01 63

Hewlett-Packard France
Zone Industrielle de Courtaboeuf
Avenue des Tropiques
F-91347 Les Ulis Cedex **ORSAY**
Tel: (6) 907-78-25
Telex: 600048F
A,CH,CM,CS,E,MP,P

Hewlett-Packard France
Paris Porte-Maillot
15, Avenue de L'Amiral Bruix
F-75782 **PARIS** CEDEX 16
Tel: 16 (1) 502-12-20
Telex: 613663F
CH,MS,P

Hewlett-Packard France
124, Boulevard Tourasse
F-64000 **PAU**
Tel: 16 (59) 80 38 02

Hewlett-Packard France
2 Allée de la Bourgonnette
F-35100 **RENNES**
Tel: 16 (99) 51-42-44
Telex: 740912F
CH,CM,E,MS,P*

Hewlett-Packard France
98 Avenue de Bretagne
F-76100 **ROUEN**
Tel: 16 (35) 63-57-66
CH**,CS

Hewlett-Packard France
4 Rue Thomas Mann
Boite Postale 56
F-67033 **STRASBOURG** Cedex
Tel: 16 (88) 28-56-46
Telex: 890141F
CH,E,MS,P*

Hewlett-Packard France
Le Péripole
20, Chemin du Pigeonnier de la Cépière
F-31083 **TOULOUSE** Cedex
Tel: 16 (61) 40-11-12
Telex: 531639F
A,CH,CS,E,P*

Hewlett-Packard France
9, rue Baudin
F-26000 **VALENCE**
Tel: 16 (75) 42 76 16

Hewlett-Packard France
Carolor
ZAC de Bois Briand
F-57640 **VIGY** (Metz)
Tel: 16 (8) 771 20 22
CH

Hewlett-Packard France
Immeuble Péricentre
F-59658 **VILLENEUVE D'ASCQ** Cedex
Tel: 16 (20) 91-41-25
Telex: 160124F
CH,E,MS,P*

GERMAN FEDERAL REPUBLIC

Hewlett-Packard GmbH
Geschäftsstelle
Keithstrasse 2-4
D-1000 **BERLIN** 30
Tel: (030) 24-90-86
Telex: 018 3405 hpbln d
A,CH,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Herrenberger Strasse 130
D-7030 **BOBLINGEN**
Tel: (7031) 14-0
Telex:
A,CH,CM,CS,E,MP,P

Hewlett-Packard GmbH
Geschäftsstelle
Emanuel-Leutze-Strasse 1
D-4000 **DUSSELDORF**
Tel: (0211) 5971-1
Telex: 085/86 533 hpdd d
A,CH,CS,E,MS,P

Hewlett-Packard GmbH
Geschäftsstelle
Schleefstr. 28a
D-4600 **DORTMUND**-Aplerbeck
Tel: (0231) 45001

Hewlett-Packard GmbH
Vertriebszentrale Frankfurt
Berner Strasse 117
Postfach 560 140
D-6000 **FRANKFURT** 56
Tel: (0611) 50-04-1
Telex: 04 13249 hpffm d
A,CH,CM,CS,E,MP,P

Hewlett-Packard GmbH
Geschäftsstelle
Aussenstelle Bad Homburg
Louisenstrasse 115
D-6380 **BAD HOMBURG**
Tel: (06172) 109-0

Hewlett-Packard GmbH
Geschäftsstelle
Kapstadtring 5
D-2000 **HAMBURG** 60
Tel: (040) 63804-1
Telex: 021 63 032 hphh d
A,CH,CS,E,MS,P

Hewlett-Packard GmbH
Geschäftsstelle
Heidering 37-39
D-3000 **HANNOVER** 61
Tel: (0511) 5706-0
Telex: 092 3259
A,CH,CM,E,MS,P

Hewlett-Packard GmbH
Geschäftsstelle
Rosslauer Weg 2-4
D-6800 **MANNHEIM**
Tel: (0621) 70050
Telex: 0462105
A,C,E

Hewlett-Packard GmbH
Geschäftsstelle
Messerschmittstrasse 7
D-7910 **NEU ULM**
Tel: 0731-70241
Telex: 0712816 HP ULM-D
A,C,E*

Hewlett-Packard GmbH
Geschäftsstelle
Ehnericherstr. 13
D-8500 **NURNBERG** 10
Tel: (0911) 5205-0
Telex 0623 860
CH,CM,E,MS,P

Hewlett-Packard GmbH
Geschäftsstelle
Eschenstrasse 5
D-8028 **TAUFKIRCHEN**
Tel: (089) 6117-1
Telex: 0524985
A,CH,CM,E,MS,P

GREAT BRITAIN

See United Kingdom

GREECE

Kostas Karaynnis S.A.
8 Omirou Street
***ATHENS** 133*
Tel: 32 30 303, 32 37 371
Telex: 215962 RKAR GR
A,CH,CM,CS,E,M,P

PLAISIO S.A.
G. Gerardos
24 Stournara Street
ATHENS
Tel: 36-11-160
Telex: 221871
P

GUATEMALA

IPESA
Avenida Reforma 3-48, Zona 9
GUATEMALA CITY
Tel: 316627, 314786
Telex: 4192 TELTRO GU
A,CH,CM,CS,E,M,P

HONG KONG

Hewlett-Packard Hong Kong, Ltd.
G.P.O. Box 795
5th Floor, Sun Hung Kai Centre
30 Harbour Road
HONG KONG
Tel: 5-8323211
Telex: 66678 HEWPA HX
Cable: HEWPACK HONG KONG
E,CH,CS,P

CET Ltd.
1402 Tung Wah Mansion
199-203 Hennessy Rd.
*Wanchia, **HONG KONG***
Tel: 5-729376
Telex: 85148 CET HX
CM

Schmidt & Co. (Hong Kong) Ltd.
Wing On Centre, 28th Floor
Connaught Road, C.
HONG KONG
Tel: 5-455644
Telex: 74766 SCHMX HX
A,M

ICELAND

Elding Trading Company Inc.
Hafnarnvoli-Tryggvagotu
P.O. Box 895
*IS-**REYKJAVIK***
Tel: 1-58-20, 1-63-03
M

INDIA

Computer products are sold through Blue Star Ltd. All computer repairs and maintenance service is done through Computer Maintenance Corp

Blue Star Ltd
Sabri Complex II Floor
24 Residency Rd
***BANGALORE** 560 025*
Tel: 55660
Telex 0845-430
Cable BLUESTAR
A,CH,CM,CS*,E*

Blue Star Ltd.
Band Box House
Prabhadevi
***BOMBAY** 400 025*
Tel: 422-3101
Telex: 011-3751
Cable: BLUESTAR
A,M

Blue Star Ltd.
Sahas
414/2 Vir Savarkar Marg
Prabhadevi
***BOMBAY** 400 025*
Tel: 422-6155
Telex: 011-4093
Cable: FROSTBLUE
A,CH,CM,CS*,E,M*

Blue Star Ltd.
Kalyan, 19 Vishwas Colony
*Alkapuri, **BORODA**, 390 005*
Tel: 65235
Cable: BLUE STAR
A

Blue Star Ltd.
7 Hare Street
***CALCUTTA** 700 001*
Tel: 12-01-31
Telex: 021-7655
Cable: BLUESTAR
A,M

Blue Star Ltd.
133 Kodambakkam High Road
***MADRAS** 600 034*
Tel: 82057
Telex: 041-379
Cable: BLUESTAR
A,M

Blue Star Ltd.
Bhandari House, 7th/8th Floors
91 Nehru Place
***NEW DELHI** 110 024*
Tel: 682547
Telex: 031-2463
Cable: BLUESTAR
A,CH,CM,CS*,E,M*

Blue Star Ltd.
15/16:C Wellesley Rd.
***PUNE** 411 011*
Tel: 22775
Cable: BLUE STAR
A

Blue Star Ltd.
2-2-47/1108 Bolarum Rd.
***SECUNDERABAD** 500 003*
Tel: 72057
Telex: 0155-459
Cable: BLUEFROST
A,E

Blue Star Ltd
T.C. 7/603 Poornima
Maruthankuzhi
***TRIVANDRUM** 695 013*
Tel: 65799
Telex 0884-259
Cable BLUESTAR
E

Computer Maintenance Corporation Ltd.
115, Sarojini Devi Road
***SECUNDERABAD** 500 003*
Tel: 310-184, 345-774
Telex: 031-2960
*CH***

SALES & SUPPORT OFFICES

Arranged alphabetically by country

INDONESIA

BERCA Indonesia P.T.
P.O.Box 496/Jkt.
Jl. Abdul Muis 62
JAKARTA
Tel: 21-373009
Telex: 46748 BERSAL IA
Cable: BERSAL JAKARTA
P

BERCA Indonesia P.T.
P.O.Box 2497/Jkt
Antara Bldg., 17th Floor
Jl. Medan Merdeka Selatan 17
JAKARTA-PUSAT
Tel: 21-344-181
Telex: BERSAL IA
A,CS,E,M

BERCA Indonesia P.T.
P.O. Box 174/SBY.
Jl. Kutei No. 11
SURABAYA
Tel: 68172
Telex: 31146 BERSAL SB
Cable: BERSAL-SURABAYA
A,E,M,P*

IRAQ

Hewlett-Packard Trading S.A.
Service Operation
Al Mansoor City 9B/3/7
BAGHDAD
Tel: 551-49-73
Telex: 212-455 HEPAIRAQ IK
CH,CS

IRELAND

Hewlett-Packard Ireland Ltd.
82/83 Lower Leeson Street
DUBLIN 2
Tel: 0001 608800
Telex: 30439
A,CH,CM,CS,E,M,P

Cardiac Services Ltd.
Kilmore Road
Artane
***DUBLIN** 5*
Tel: (01) 351820
Telex: 30439
M

ISRAEL

Eldan Electronic Instrument Ltd.
P.O.Box 1270
***JERUSALEM** 91000*
16, Ohaliav St.
***JERUSALEM** 94467*
Tel: 533 221, 553 242
Telex: 25231 AB/PAKRD IL
A

Electronics Engineering Division
Motorola Israel Ltd.
16 Kremenetski Street
P.O. Box 25016
TEL-AVIV 67899
Tel: 3 88 388
Telex: 33569 Motil IL
Cable: BASTEL Tel-Aviv
CH,CM,CS,E,M,P

ITALY

Hewlett-Packard Italiana S.p.A
Traversa 99C
Via Giulio Petroni, 19
I-70124 **BARI**
Tel: (080) 41-07-44
M

Hewlett-Packard Italiana S.p.A.
Via Martin Luther King, 38/III
I-40132 **BOLOGNA**
Tel: (051) 402394
Telex: 511630
CH,E,MS

Hewlett-Packard Italiana S.p.A.
Via Principe Nicola 43G/C
I-95126 **CATANIA**
Tel: (095) 37-10-87
Telex: 970291
C,P

Hewlett-Packard Italiana S.p.A.
Via G. Di Vittorio 9
I-20063 **CERNUSCO SUL NAVIGLIO**
(Milano)
Tel: (02) 923691
Telex: 334632
A,CH,CM,CS,E,MP,P

Hewlett-Packard Italiana S.p.A.
Via C. Colombo 49
I-20090 **TREZZANO SUL NAVIGLIO**
(Milano)
Tel: (02) 4459041
Telex: 322116
C,M

Hewlett-Packard Italiana S.p.A.
Via Nuova San Rocco a
Capodimonte, 62/A
I-80131 **NAPOLI**
Tel: (081) 7413544
Telex: 710698
A,CH,E

Hewlett-Packard Italiana S.p.A.
Viale G. Modugno 33
I-16156 **GENOVA PEGLI**
Tel: (010) 68-37-07
Telex: 215238
E,C

Hewlett-Packard Italiana S.p.A.
Via Pelizzo 15
I-35128 **PADOVA**
Tel: (049) 664888
Telex: 430315
A,CH,E,MS

Hewlett-Packard Italiana S.p.A.
Viale C. Pavese 340
I-00144 **ROMA EUR**
Tel: (06) 54831
Telex: 610514
A,CH,CM,CS,E,MS,P*

Hewlett-Packard Italiana S.p.A.
Via di Casellina 57/C
I-50018 **SCANDICCI-FIRENZE**
Tel: (055) 753863

Hewlett-Packard Italiana S.p.A.
Corso Svizzera, 185
I-10144 **TORINO**
Tel: (011) 74 4044
Telex: 221079
CH,E

JAPAN

Yokogawa-Hewlett-Packard Ltd.
152-1, Onna
ATSUGI, Kanagawa, 243
Tel: (0462) 28-0451
CM,C*,E

Yokogawa-Helwett-Packard Ltd.
Meiji-Seimei Bldg. 6F
3-1 Hon Chiba-Cho
CHIBA, 280
Tel: 472 25 7701
E,CH,CS

Yokogawa-Hewlett-Packard Ltd.
Yasuda-Seimei Hiroshima Bldg.
6-11, Hon-dori, Naka-ku
HIROSHIMA, 730
Tel: 82-241-0611

Yokogawa-Hewlett-Packard Ltd.
Towa Building
2-3, Kaigan-dori, 2 Chome Chuo-ku
KOBE, 650
Tel: (078) 392-4791
C,E

Yokogawa-Hewlett-Packard Ltd.
Kumagaya Asahi 82 Bldg
3-4 Tsukuba
KUMAGAYA, Saitama 360
Tel: (0485) 24-6563
CH,CM,E

Yokogawa-Hewlett-Packard Ltd.
Asahi Shinbun Daiichi Seimei Bldg.
4-7, Hanabata-cho
KUMAMOTO,860
Tel: (0963) 54-7311
CH,E

Yokogawa-Hewlett-Packard Ltd.
Shin-Kyoto Center Bldg.
614, Higashi-Shiokoji-cho
Karasuma-Nishiiru
Shiokoji-dori, Shimogyo-ku
KYOTO, 600
Tel: 075-343-0921
CH,E

Yokogawa-Hewlett-Packard Ltd.
Mito Mitsui Bldg
4-73, Sanno-maru, 1 Chome
MITO, Ibaraki 310
Tel: (0292) 25-7470
CH,CM,E

Yokogawa-Hewlett-Packard Ltd.
Sumitomo Seimei 14-9 Bldg.
Meieki-Minami, 2 Chome
Nakamura-ku
NAGOYA, 450
Tel: (052) 571-5171
CH,CM,CS,E,MS

Yokogawa-Hewlett-Packard Ltd.
Chuo Bldg.,
4-20 Nishinakajima, 5 Chome
Yodogawa-ku
OSAKA, 532
Tel: (06) 304-6021
Telex: YHPOSA 523-3624
A,CH,CM,CS,E,MP,P*

Yokogawa-Hewlett-Packard Ltd.
27-15, Yabe, 1 Chome
SAGAMIHARA Kanagawa, 229
Tel: 0427 59-1311

Yokogawa-Hewlett-Packard Ltd.
Daiichi Seimei Bldg.
7-1, Nishi Shinjuku, 2 Chome
Shinjuku-ku,**TOKYO** 160
Tel: 03-348-4611
CH,E

Yokogawa-Hewlett-Packard Ltd.
29-21 Takaido-Higashi, 3 Chome
Suginami-ku **TOKYO** 168
Tel: (03) 331-611
Telex: 232-2024 YHPTOK
A,CH,CM,CS,E,MP,P*

Yokogawa-Hewlett-Packard Ltd.
Daiichi Asano Building
2-8, Odori, 5 Chome
UTSUNOMIYA, Tochigi 320
Tel: (0286) 25-7155
CH,CS,E

Yokogawa-Hewlett-Packard Ltd.
Yasuda Seimei Nishiguchi Bldg.
30-4 Tsuruya-cho, 3 Chome
YOKOHAMA 221
Tel: (045) 312-1252
CH,CM,E

JORDAN

Mouasher Cousins Company
P.O. Box 1387
AMMAN
Tel: 24907, 39907
Telex: 21456 SABCO JO
CH,E,M,P

KENYA

ADCOM Ltd., Inc., Kenya
P.O.Box 30070
NAIROBI
Tel: 331955
Telex: 22639
E,M

KOREA

Samsung Electronics HP Division
12 Fl. Kinam Bldg.
San 75-31, Yeoksam-Dong
Kangnam-Ku
Yeongdong P.O. Box 72
SEOUL
Tel: 555-7555, 555-5447
Telex: K27364 SAMSAN
A,CH,CM,CS,E,M,P

KUWAIT

Al-Khaldiya Trading & Contracting
P.O. Box 830 Safat
KUWAIT
Tel: 42-4910, 41-1726
Telex: 22481 Areeg kt
CH,E,M

Photo & Cine Equipment
P.O. Box 270 Safat
KUWAIT
Tel: 42-2846, 42-3801
Telex: 22247 Matin kt
P

LEBANON

G.M. Dolmadjian
Achrafieh
P.O. Box 165.167
BEIRUT
Tel: 290293
*MP**
Computer Information Systems
P.O. Box 11-6274
BEIRUT
Tel: 89 40 73
Telex: 22259
C

LUXEMBOURG

Hewlett-Packard Belgium S.A./N.V.
Blvd de la Woluwe, 100
Woluwedal
B-1200 **BRUSSELS**
Tel: (02) 762-32-00
Telex: 23-494 paloben bru
A,CH,CM,CS,E,MP,P

MALAYSIA

Hewlett-Packard Sales (Malaysia)
Sdn. Bhd.
1st Floor, Bangunan British
American
Jalan Semantan, Damansara Heights
KUALA LUMPUR 23-03
Tel: 943022
Telex: MA31011
A,CH,E,M,P*

SALES & SUPPORT OFFICES

Arranged alphabetically by country

MAYLAYSIA (Cont'd)
Protel Engineering
P.O.Box 1917
Lot 6624, Section 64
23/4 Pending Road
Kuching, ***SARAWAK***
Tel: 36299
Telex: MA 70904 PROMAL
Cable: PROTELENG
A,E,M

MALTA
Philip Toledo Ltd.
Notabile Rd.
MRIEHEL
Tel: 447 47, 455 66
Telex: Media MW 649
E,P

MEXICO
Hewlett-Packard Mexicana, S.A. de C.V.
Av. Periferico Sur No. 6501
Tepepan, Xochimilco
16020 **MEXICO D.F.**
Tel: 6-76-46-00
Telex: 17-74-507 HEWPACK MEX
A,CH,CS,E,MS,P

Hewlett-Packard Mexicana, S.A. de C.V.
Ave. Colonia del Valle 409
Col. del Valle
Municipio de Garza Garcia
MONTERREY, Nuevo Leon
Tel: 78 42 41
Telex: 038 410
CH

ECISA
José Vasconcelos No. 218
Col. Condesa Deleg. Cuauhtémoc
***MEXICO** D.F. 06140*
Tel: 553-1206
Telex: 17-72755 ECE ME
M

MOROCCO
Dolbeau
81 rue Karatchi
CASABLANCA
Tel: 3041-82, 3068-38
Telex: 23051, 22822
E

Gerep
2 rue d'Agadir
Boite Postale 156
CASABLANCA
Tel: 272093, 272095
Telex: 23 739
P

NETHERLANDS
Hewlett-Packard Nederland B.V.
Van Heuven Goedhartlaan 121
NL 1181KK **AMSTELVEEN**
P.O. Box 667
NL 1180 AR **AMSTELVEEN**
Tel: (020) 47-20-21
Telex: 13 216 HEPA NL
A,CH,CM,CS,E,MP,P

Hewlett-Packard Nederland B.V.
Bongerd 2
NL 2906VK **CAPELLE A/D IJSSEL**
P.O Box 41
NL 2900AA **CAPELLE A D IJSSEL**
Tel: (10) 51-64-44
Telex: 21261 HEPAC NL
A,CH,CS,E

Hewlett-Packard Nederland B.V.
Pastoor Petersstraat 134-136
NL 5612 LV **EINDHOVEN**
P.O. Box 2342
NL 5600 CH **EINDHOVEN**
Tel: (040) 326911
Telex: 51484 hepae nl
A,CH**,E,M

NEW ZEALAND
Hewlett-Packard (N.Z.) Ltd.
5 Owens Road
P.O. Box 26-189
Epsom, **AUCKLAND**
Tel: 687-159
Cable: HEWPACK Auckland
CH,CM,E,P*

Hewlett-Packard (N.Z.) Ltd.
4-12 Cruickshank Street
Kilbirnie, **WELLINGTON 3**
P.O. Box 9443
Courtenay Place, **WELLINGTON 3**
Tel: 877-199
Cable: HEWPACK Wellington
CH,CM,E,P

Northrop Instruments & Systems Ltd.
369 Khyber Pass Road
P.O. Box 8602
AUCKLAND
Tel: 794-091
Telex: 60605
A,M

Northrop Instruments & Systems Ltd.
110 Mandeville St.
P.O. Box 8388
CHRISTCHURCH
Tel: 486-928
Telex: 4203
A,M

Northrop Instruments & Systems Ltd.
Sturdee House
85-87 Ghuznee Street
P.O. Box 2406
WELLINGTON
Tel: 850-091
Telex: NZ 3380
A,M

NORTHERN IRELAND
See United Kingdom

NORWAY
Hewlett-Packard Norge A/S
Folke Bernadottes vei 50
P.O. Box 3558
N-5033 **FYLLINGSDALEN** (Bergen)
Tel: 0047/5/16 55 40
Telex: 16621 hpnas n
CH,CS,E,MS

Hewlett-Packard Norge A/S
Österndalen 16-18
P.O. Box 34
N-1345 **OSTERÅS**
Tel: 0047/2/17 11 80
Telex: 16621 hpnas n
A,CH,CM,CS,E,M,P

OMAN
Khimjil Ramdas
P.O. Box 19
MUSCAT
Tel: 722225, 745601
Telex 3289 BROKER MB MUSCAT
P

Suhail & Saud Bahwan
P.O Box 169
MUSCAT
Tel: 734 201-3
Telex 3274 BAHWAN MB

PAKISTAN
Mushko & Company Ltd.
1-B, Street 43
Sector F-8/1
ISLAMABAD
Tel: 51071
Cable: FEMUS Rawalpindi
A,E,M

Mushko & Company Ltd.
Oosman Chambers
Abdullah Haroon Road
***KARACHI** 0302*
Tel: 524131, 524132
Telex: 2894 MUSKO PK
Cable: COOPERATOR Karachi
*A,E,M,P**

PANAMA
Electrónico Balboa, S.A.
Calle Samuel Lewis, Ed. Alfa
Apartado 4929
PANAMA 5
Tel: 63-6613, 63-6748
Telex: 3483 ELECTRON PG
A,CM,E,M,P

PERU
Cía Electro Médica S.A.
Los Flamencos 145, San Isidro
Casilla 1030
***LIMA** 1*
Tel: 41-4325, 41-3703
Telex: Pub. Booth 25306
CM,E,M,P

PHILIPPINES
The Online Advanced Systems Corporation
Rico House, Amorsolo Cor. Herrera Street
Legaspi Village, Makati
P.O. Box 1510
Metro ***MANILA***
Tel: 85-35-81, 85-34-91, 85-32-21
Telex: 3274 ONLINE
A,CH,CS,E,M

Electronic Specialists and Proponents Inc.
690-B Epifanio de los Santos Avenue
Cubao, ***QUEZON CITY***
P.O. Box 2649 Manila
Tel: 98-96-81, 98-96-82, 98-96-83
Telex: 40018, 42000 ITT GLOBE MACKAY BOOTH
P

PORTUGAL
Mundinter
Intercambio Mundial de Comércio S.A.R.L.
P.O. Box 2761
Av. Antonio Augusto de Aguiar 138
P-LISBON
Tel: (19) 53-21-31, 53-21-37
Telex: 16691 munter p
M

Soquimica
Av. da Liberdade, 220-2
1298 ***LISBOA** Codex*
Tel: 56 21 81/2/3
Telex: 13316 SABASA
P

Telectra-Empresa Técnica de Equipmentos Eléctricos S.A.R.L
Rua Rodrigo da Fonseca 103
P.O. Box 2531
***P-LISBON** 1*
Tel: (19) 68-60-72
Telex: 12598
CH,CS,E,P

PUERTO RICO
Hewlett-Packard Puerto Rico
Ave. Muñoz Rivera #101
Esq. Calle Ochoa
HATO REY, Puerto Rico 00918
Tel: (809) 754-7800

Hewlett-Packard Puerto Rico
Calle 272 Edificio 203
Urb. Country Club
RIO PIEDRAS, Puerto Rico
P.O. Box 4407
CAROLINA, Puerto Rico 00628
Tel: (809) 762-7255
A,CH,CS

QATAR
Computearbia
P.O. Box 2750
DOHA
Tel: 883555
Telex: 4806 CHPARB
P

Eastern Technical Services
P.O.Box 4747
DOHA
Tel: 329 993
Telex: 4156 EASTEC DH

Nasser Trading & Contracting
P.O.Box 1563
DOHA
Tel: 22170, 23539
Telex: 4439 NASSER DH
M

SAUDI ARABIA
Modern Electronic Establishment
Hewlett-Packard Division
P.O. Box 22015
Thuobah
AL-KHOBAR
Tel: 895-1760, 895-1764
Telex: 671 106 HPMEEK SJ
Cable: ELECTA AL-KHOBAR
CH,CS,E,M

Modern Electronic Establishment
Hewlett-Packard Division
P.O. Box 1228
Redec Plaza, 6th Floor
JEDDAH
Tel: 644 38 48
Telex: 4027 12 FARNAS SJ
Cable: ELECTA JEDDAH
CH,CS,E,M

Modern Electronic Establishment
Hewlett-Packard Division
P.O Box 22015
RIYADH
Tel: 491-97 15, 491-63 87
Telex: 202049 MEERYD SJ
CH,CS,E,M

Abdul Ghani El Ajou
P.O Box 78
RIYADH
Tel. 40 41 717
Telex: 200 932 EL AJOU
P

SCOTLAND
See United Kingdom

SINGAPORE
Hewlett-Packard Singapore (Sales)
Pte Ltd
#08-00 Inchcape House
450-2 Alexandra Road
P.O. Box 58 Alexandra Rd Post Office
SINGAPORE, 9115
Tel 631788
Telex: HPSGSO RS 34209
Cable HEWPACK, Singapore

SALES & SUPPORT OFFICES

Arranged alphabetically by country

SINGAPORE (Cont'd)

Dynamar International Ltd.
Unit 05-11 Block 6
Kolam Ayer Industrial Estate
***SINGAPORE** 1334*
Tel: 747-6188
Telex: RS 26283
CM

SOUTH AFRICA

Hewlett-Packard So Africa (Pty.) Ltd.
P.O. Box 120
Howard Place **CAPE PROVINCE** 7450
Pine Park Center, Forest Drive,
Pinelands
CAPE PROVINCE 7405
Tel: 53-7954
Telex: 57-20006
A,CH,CM,E,MS,P

Hewlett-Packard So Africa (Pty.) Ltd.
P.O. Box 37099
92 Overport Drive
DURBAN 4067
Tel: 28-4178, 28-4179, 28-4110
Telex: 6-22954
CH,CM

Hewlett-Packard So Africa (Pty.) Ltd.
6 Linton Arcade
511 Cape Road
Linton Grange
PORT ELIZABETH 6000
Tel: 041-302148
CH

Hewlett-Packard So Africa (Pty.) Ltd.
P.O.Box 33345
Glenstantia 0010 **TRANSVAAL**
1st Floor East
Constantia Park Ridge Shopping
Centre
Constantia Park
PRETORIA
Tel: 982043
Telex: 32163
CH,E

Hewlett-Packard So Africa (Pty.) Ltd.
Private Bag Wendywood
SANDTON 2144
Tel: 802-5111, 802-5125
Telex: 4-20877
Cable: HEWPACK Johannesburg
A,CH,CM,CS,E,MS,P

SPAIN

Hewlett-Packard Española S.A.
Calle Entenza, 321
E-BARCELONA 29
Tel: 322.24.51, 321.73.54
Telex: 52603 hpbee
A,CH,CS,E,MS,P

Hewlett-Packard Española S.A.
Calle San Vicente S/No
Edificio Albia II
E-BILBAO 1
Tel: 423.83.06
A,CH,E,MS

Hewlett-Packard Española S.A.
Crta. de la Coruña, Km. 16, 400
Las Rozas
E-MADRID
Tel: (1) 637.00.11
CH,CS,M

Hewlett-Packard Española S.A.
Avda. S. Francisco Javier, S/no
Planta 10. Edificio Sevilla 2,
E-SEVILLA 5
Tel: 64.44.54
Telex: 72933
A,CS,MS,P

Hewlett-Packard Española S.A.
Calle Ramon Gordillo, 1 (Entlo.3)
E-VALENCIA 10
Tel: 361-1354
CH,P

SWEDEN

Hewlett-Packard Sverige AB
Sunnanvagen 14K
S-22226 **LUND**
Tel: (046) 13-69-79
Telex: (854) 17886 (via Spånga
office)
CH

Hewlett-Packard Sverige AB
Östra Tullgatan 3
S-21128 **MALMO**
Tel: (040) 70270
Telex: (854) 17886 (via Spånga
office)

Hewlett-Packard Sverige AB
Västra Vintergatan 9
S-70344 **ÖREBRO**
Tel: (19) 10-48-80
Telex: (854) 17886 (via Spånga
office)
CH

Hewlett-Packard Sverige AB
Skalholtsgatan 9, Kista
Box 19
S-16393 **SPÅNGA**
Tel: (08) 750-2000
Telex: (854) 17886
Telefax: (08) 7527781
A,CH,CM,CS,E,MS,P

Hewlett-Packard Sverige AB
Frötallisgatan 30
S-42132 **VÄSTRA-FRÖLUNDA**
Tel: (031) 49-09-50
Telex: (854) 17886 (via Spånga
office)
CH,E,P

SWITZERLAND

Hewlett-Packard (Schweiz) AG
Clarastrasse 12
CH-4058 **BASEL**
Tel: (61) 33-59-20
A

Hewlett-Packard (Schweiz) AG
7, rue du Bois-du-Lan
Case Postale 365
CH-1217 **MEYRIN** 2
Tel: (0041) 22-83-11-11
Telex:27333 HPAG CH
CH,CM,CS

Hewlett-Packard (Schweiz) AG
Allmend 2
CH-8967 **WIDEN**
Tel: (0041) 57 31 21 11
Telex: 53933 hpag ch
Cable: HPAG CH
A,CH,CM,CS,E,MS,P

SYRIA

General Electronic Inc.
Nuri Basha Ahnaf Ebn Kays Street
P.O. Box 5781
DAMASCUS
Tel: 33-24-87
Telex: 411 215
Cable: ELECTROBOR DAMASCUS
E

Middle East Electronics
P.O.Box 2308
Abu Rumnaneh
DAMASCUS
Tel: 33 4 5 92
Telex: 411 304
M

TAIWAN

Hewlett-Packard Far East Ltd.
Kaohsiung Office
2/F 68-2, Chung Cheng 3rd Road
KAOHSIUNG
Tel: (07) 241-2318
CH,CS,E

Hewlett-Packard Far East Ltd.
Taiwan Branch
8th Floor
337 Fu Hsing North Road
TAIPEI
Tel: (02) 712-0404
Telex: 24439 HEWPACK
Cable:HEWPACK Taipei
A,CH,CM,CS,E,M,P

Ing Lih Trading Co.
3rd Floor, 7 Jen-Ai Road, Sec. 2
TAIPEI 100
Tel: (02) 3948191
Cable: INGLIH TAIPEI
A

THAILAND

Unimesa
30 Patpong Ave., Suriwong
***BANGKOK** 5*
Tel: 235-5727
Telex: 84439 Simonco TH
Cable: UNIMESA Bangkok
A,CH,CS,E,M

Bangkok Business Equipment Ltd.
5/5-6 Dejo Road
BANGKOK
Tel: 234-8670, 234-8671
Telex: 87669-BEQUIPT TH
Cable: BUSIQUIPT Bangkok
P

TRINIDAD & TOBAGO

Caribbean Telecoms Ltd.
50/A Jerningham Avenue
P.O. Box 732
PORT-OF-SPAIN
Tel: 62-44213, 62-44214
Telex: 235,272 HUGCO WG
CM,E,M,P

TUNISIA

Tunisie Electronique
31 Avenue de la Liberte
TUNIS
Tel: 280-144
E,P

Corema
1 ter. Av. de Carthage
TUNIS
Tel: 253-821
Telex: 12319 CABAM TN
M

TURKEY

Teknim Company Ltd.
Iran Caddesi No. 7
*Kavaklidere, **ANKARA***
Tel: 275800
Telex: 42155 TKNM TR
E

E.M.A.
Medina Eldem Sokak No.41/6
Yuksel Caddesi
ANKARA
Tel: 175 622
Telex: 42 591
M

UNITED ARAB EMIRATES

Emitac Ltd.
P.O. Box 2711
ABU DHABI
Tel: 82 04 19-20
Cable: EMITAC ABUDHABI

Emitac Ltd.
P.O. Box 1641
SHARJAH
Tel: 591 181
Telex: 68136 Emitac Sh
CH,CS,E,M,P

UNITED KINGDOM

GREAT BRITAIN

Hewlett-Packard Ltd.
Trafalgar House
Navigation Road
ALTRINCHAM
Cheshire WA14 1NU
Tel: 061 928 6422
Telex: 668068
A,CH,CS,E,M,MS,P

Hewlett-Packard Ltd.
Elstree House, Elstree Way
BOREHAMWOOD, Herts WD6 1SG
Tel: 01 207 5000
Telex: 8952716
E,CH,CS,P

Hewlett-Packard Ltd.
Oakfield House, Oakfield Grove
Clifton **BRISTOL**, Avon BS8 2BN
Tel: 0272 736806
Telex: 444302
CH,CS,E,P

Hewlett-Packard Ltd.
Bridewell House
Bridewell Place
LONDON EC4V 6BS
Tel: 01 583 6565
Telex: 298163
CH,CS,P

Hewlett-Packard Ltd.
Fourier House
257-263 High Street
LONDON COLNEY
Herts. AL2 1HA, St. Albans
Tel: 0727 24400
Telex: 1-8952716
CH,CS

Hewlett-Packard Ltd.
Pontefract Road
NORMANTON, West Yorkshire WF6 1RN
Tel: 0924 895566
Telex: 557355
CH,CS,P

Hewlett-Packard Ltd.
The Quadrangle
106-118 Station Road
REDHILL, Surrey RH1 1PS
Tel: 0737 68655
Telex: 947234
CH,CS,E,P

SALES & SUPPORT OFFICES

Arranged alphabetically by country

GREAT BRITAIN (Cont'd)
Hewlett-Packard Ltd.
Avon House
435 Stratford Road
Shirley, **SOLIHULL**, West Midlands
B90 4BL
Tel: 021 745 8800
Telex: 339105
CH,CS,E,P

Hewlett-Packard Ltd.
West End House
41 High Street, West End
SOUTHAMPTON
Hampshire S03 3DQ
Tel: 04218 6767
Telex: 477138
CH,CS,P

Hewlett-Packard Ltd.
Eskdale Rd.
Winnersh, **WOKINGHAM**
Berkshire RG11 5DZ
Tel: 0734 696622
Telex: 848884
E

Hewlett-Packard Ltd.
King Street Lane
Winnersh, **WOKINGHAM**
Berkshire RG11 5AR
Tel: 0734 784774
Telex: 847178
A,CH,CS,E,M,MP,P

Hewlett-Packard Ltd.
Nine Mile Ride
Easthampstead, WOKINGHAM
Berkshire, 3RG11 3LL
Tel: 0344 773100
Telex: 848805
CH,CS,E,P

IRELAND

NORTHERN IRELAND
Hewlett-Packard Ltd.
Cardiac Services Building
95A Finaghy Road South
BELFAST BT10 0BY
Tel: 0232 625-566
Telex: 747626
CH,CS

SCOTLAND
Hewlett-Packard Ltd.
SOUTH QUEENSFERRY
West Lothian, EH30 9TG
Tel: 031 331 1188
Telex: 72682
CH,CM,CS,E,M,P

UNITED STATES

Alabama
Hewlett-Packard Co.
700 Century Park South, Suite 128
BIRMINGHAM, AL 35226
Tel: (205) 822-6802
A,CH,M

Hewlett-Packard Co
420 Wynn Drive
HUNTSVILLE, AL 35805
P.O. Box 7700
HUNTSVILLE, AL 35807
Tel: (205) 830-2000
CH,CM,CS,E,M*

Arizona
Hewlett-Packard Co
8080 Pointe Parkway West
PHOENIX, AZ 85044
Tel: (602) 273-8000
A,CH,CM,CS,E,MS

Hewlett-Packard Co.
2424 East Aragon Road
TUCSON, AZ 85706
Tel: (602) 889-4631
CH,E,MS**

California
Hewlett-Packard Co.
99 South Hill Dr.
BRISBANE, CA 94005
Tel: (415) 330-2500
CH,CS

Hewlett-Packard Co.
P.O. Box 7830 (93747)
5060 E. Clinton Avenue, Suite 102
FRESNO, CA 93727
Tel: (209) 252-9652
CH,CS,MS

Hewlett-Packard Co.
P.O. Box 4230
1430 East Orangethorpe
FULLERTON, CA 92631
Tel: (714) 870-1000
CH,CM,CS,E,MP

Hewlett-Packard Co.
320 S. Kellogg, Suite B
GOLETA, CA 93117
Tel: (805) 967-3405
CH

Hewlett-Packard Co.
5400 W. Rosecrans Boulevard
LAWNDALE, CA 90260
P.O. Box 92105
LOS ANGELES, CA 90009
Tel: (213) 970-7500
Telex: 910-325-6608
CH,CM,CS,MP

Hewlett-Packard Co.
3155 Porter Oaks Drive
PALO ALTO, CA 94304
Tel: (415) 857-8000
CH,CS,E

Hewlett-Packard Co.
4244 So. Market Court, Suite A
P.O. Box 15976
SACRAMENTO, CA 95852
Tel: (916) 929-7222
A*,CH,CS,E,MS

Hewlett-Packard Co.
9606 Aero Drive
P.O. Box 23333
SAN DIEGO, CA 92139
Tel: (619) 279-3200
CH,CM,CS,E,MP

Hewlett-Packard Co.
2305 Camino Ramon "C"
SAN RAMON, CA 94583
Tel: (415) 838-5900
CH,CS

Hewlett-Packard Co
3005 Scott Boulevard
SANTA CLARA, CA 95050
Tel: (408) 988-7000
Telex: 910-338-0586
A,CH,CM,CS,E,MP

Hewlett-Packard Co.
5703 Corsa Avenue
WESTLAKE VILLAGE, CA 91362
Tel: (213) 706-6800
E*,CH*,CS*

Colorado
Hewlett-Packard Co
24 Inverness Place, East
ENGLEWOOD, CO 80112
Tel: (303) 649-5000
A,CH,CM,CS,E,MS

Connecticut
Hewlett-Packard Co.
47 Barnes Industrial Road South
P.O. Box 5007
WALLINGFORD, CT 06492
Tel: (203) 265-7801
A,CH,CM,CS,E,MS

Florida
Hewlett-Packard Co.
2901 N.W. 62nd Street
P.O. Box 24210
FORT LAUDERDALE, FL 33307
Tel: (305) 973-2600
CH,CS,E,MP

Hewlett-Packard Co.
6177 Lake Ellenor Drive
P.O. Box 13910
ORLANDO, FL 32859
Tel: (305) 859-2900
A,CH,CM,CS,E,MS

Hewlett-Packard Co.
5750B N. Hoover Blvd., Suite 123
P.O. Box 15200
TAMPA, FL 33614
Tel: (813) 884-3282
A*,CH,CM,CS,E*,M*

Georgia
Hewlett-Packard Co.
2000 South Park Place
P.O. Box 105005
ATLANTA, GA 30348
Tel: (404) 955-1500
Telex: 810-766-4890
A,CH,CM,CS,E,MP

Hawaii
Hewlett-Packard Co.
Kawaiahao Plaza, Suite 190
567 South King Street
HONOLULU, HI 96813
Tel: (808) 526-1555
A,CH,E,MS

Illinois
Hewlett-Packard Co.
304 Eldorado Road
P.O. Box 1607
BLOOMINGTON, IL 61701
Tel: (309) 662-9411
CH,MS**

Hewlett-Packard Co.
1100 31st Street, Suite 100
DOWNERS GROVE, IL 60515
Tel: (312) 960-5760
CH,CS

Hewlett-Packard Co.
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800
Telex: 910-687-1066
A,CH,CM,CS,E,MP

Indiana
Hewlett-Packard Co
7301 No. Shadeland Avenue
P.O. Box 50807
INDIANAPOLIS, IN 46250
Tel: (317) 842-1000
A,CH,CM,CS,E,MS

Iowa
Hewlett-Packard Co
1776 22nd Street, Suite 1
WEST DES MOINES, IA 50265
Tel: (515) 224-1435
CH,MS**

Kansas
Hewlett-Packard Co.
7804 East Funston Road, #203
WICHITA, KS 67207
Tel: (316) 684-8491
CH

Kentucky
Hewlett-Packard Co.
10300 Linn Station Road, #100
LOUISVILLE, KY 40223
Tel: (502) 426-0100
A,CH,CS,MS

Louisiana
Hewlett-Packard Co.
160 James Drive East
ST. ROSE, LA 70087
P.O. Box 1449
KENNER, LA 70063
Tel: (504) 467-4100
A,CH,CS,E,MS

Maryland
Hewlett-Packard Co
3701 Koppers Street
BALTIMORE, MD 21227
Tel: (301) 644-5800
Telex: 710-862-1943
A,CH,CM,CS,E,MS

Hewlett-Packard Co.
2 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 948-6370
A,CH,CM,CS,E,MP

Massachusetts
Hewlett-Packard Co
1775 Minuteman Road
ANDOVER, MA 01810
Tel: (617) 682-1500
A,C,CH,CS,CM,E,MP,P*

Hewlett-Packard Co
32 Hartwell Avenue
LEXINGTON, MA 02173
Tel: (617) 861-8960
CH,CS,E

Michigan
Hewlett-Packard Co
4326 Cascade Road S.E
GRAND RAPIDS, MI 49506
Tel: (616) 957-1970
CH,CS,MS

Hewlett-Packard Co
1771 W. Big Beaver Road
TROY, MI 48084
Tel: (313) 643-6474
CH,CS

Minnesota
Hewlett-Packard Co
2025 W. Larpenteur Ave
ST. PAUL, MN 55113
Tel: (612) 644-1100
A,CH,CM,CS,E,MP

Missouri
Hewlett-Packard Co
11131 Colorado Avenue
KANSAS CITY, MO 64137
Tel (816) 763-8000
A,CH,CM,CS,E,MS

Hewlett-Packard Co
13001 Hollenberg Drive
BRIDGETON, MO 63044
Tel (314) 344-5100
A,CH,CS,E,MP

SALES & SUPPORT OFFICES

Arranged alphabetically by country

UNITED STATES (Cont'd)

Nebraska
Hewlett-Packard
10824 Old Mill Rd., Suite 3
OMAHA, NE 68154
Tel: (402) 334-1813
CM,MS

New Jersey
Hewlett-Packard Co.
120 W. Century Road
PARAMUS, NJ 07652
Tel: (201) 265-5000
A,CH,CM,CS,E,MP

Hewlett-Packard Co.
60 New England Av. West
PISCATAWAY, NJ 08854
Tel: (201) 981-1199
A,CH,CM,CS,E

New Mexico
Hewlett-Packard Co.
11300 Lomas Blvd.,N.E.
P.O. Box 11634
ALBUQUERQUE, NM 87112
Tel: (505)* 292-1330
CH,CS,E,MS

New York
Hewlett-Packard Co.
5 Computer Drive South
ALBANY, NY 12205
Tel: (518) 458-1550
A,CH,E,MS

Hewlett-Packard Co.
9600 Main Street
P.O. Box AC
CLARENCE, NY 14031
Tel: (716) 759-8621
CH

Hewlett-Packard Co.
200 Cross Keys Office Park
FAIRPORT, NY 14450
Tel: (716) 223-9950
CH,CM,CS,E,MS

Hewlett-Packard Co.
7641 Henry Clay Blvd.
LIVERPOOL, NY 13088
Tel: (315) 451-1820
A,CH,CM,E,MS

Hewlett-Packard Co.
No. 1 Pennsylvania Plaza
55th Floor
34th Street & 8th Avenue
MANHATTAN NY 10119
Tel: (212) 971-0800
CH,CS,E*,M*

Hewlett-Packard Co.
250 Westchester Avenue
WHITE PLAINS, NY 10604
Tel: (914) 684-6100
CM,CH,CS,E

Hewlett-Packard Co.
3 Crossways Park West
WOODBURY, NY 11797
Tel: (516) 921-0300
A,CH,CM,CS,E,MS

North Carolina
Hewlett-Packard Co.
5605 Roanne Way
P.O. Box 26500
GREENSBORO, NC 27420
Tel: (919) 852-1800
A,CH,CM,CS,E,MS

Ohio
Hewlett-Packard Co.
9920 Carver Road
CINCINNATI, OH 45242
Tel: (513) 891-9870
CH,CS,MS

Hewlett-Packard Co.
16500 Sprague Road
CLEVELAND, OH 44130
Tel: (216) 243-7300
A,CH,CM,CS,E,MS

Hewlett-Packard Co.
962 Crupper Ave.
COLUMBUS, OH 43229
Tel: (614) 436-1041
Eff: Nov. 25, 1983
675 Brooksedge Blvd
WESTERVILLE, OH 43081
CH,CM,CS,E*

Hewlett-Packard Co.
330 Progress Rd.
DAYTON, OH 45449
Tel: (513) 859-8202
A,CH,CM,E*,MS

Oklahoma
Hewlett-Packard Co.
304 N. Meridian, Suite A
P.O. Box 75609
OKLAHOMA CITY, OK 73147
Tel: (405) 946-9499
A*,CH,E*,MS

Hewlett-Packard Co.
3840 S. 103rd E. Avenue, #100
P.O. Box 35747
TULSA, OK 74153
Tel: (918) 665-3300
A**,CH,CS,M*

Oregon
Hewlett-Packard Co.
9255 S. W. Pioneer Court
P.O. Box 328
WILSONVILLE, OR 97070
Tel: (503) 682-8000
A,CH,CS,E*,MS

Pennsylvania
Hewlett-Packard Co.
111 Zeta Drive
PITTSBURGH, PA 15238
Tel: (412) 782-0400
A,CH,CS,E,MP

Hewlett-Packard Co.
2750 Monroe Boulevard
P.O. Box 713
VALLEY FORGE, PA 19482
Tel: (215) 666-9000
A,CH,CM,E,M

South Carolina
Hewlett-Packard Co.
Brookside Park, Suite 122
1 Harbison Way
P.O. Box 21708
COLUMBIA, SC 29221
Tel: (803) 732-0400
CH,E,MS

Hewlett-Packard Co.
Koger Executive Center
Chesterfield Bldg., Suite 124
GREENVILLE, SC 29615
Tel: (803) 297-4120

Tennessee
Hewlett-Packard Co.
224 Peters Road, Suite 102
P.O. Box 22490
KNOXVILLE, TN 37922
Tel: (615) 691-2371
A*,CH,MS

Hewlett-Packard Co.
3070 Directors Row
MEMPHIS, TN 38131
Tel: (901) 346-8370
A,CH,MS

Texas
Hewlett-Packard Co.
4171 North Mesa
Suite C-110
EL PASO, TX 79902
Tel: (915) 533-3555
CH,E*,MS**

Hewlett-Packard Co.
10535 Harwin Drive
P.O. Box 42816
HOUSTON, TX 77042
Tel: (713) 776-6400
A,CH,CM,CS,E,MP

Hewlett-Packard Co.
930 E. Campbell Rd.
P.O. Box 1270
RICHARDSON, TX 75080
Tel: (214) 231-6101
A,CH,CM,CS,E,MP

Hewlett-Packard Co.
1020 Central Parkway South
P.O. Box 32993
SAN ANTONIO, TX 78216
Tel: (512) 494-9336
CH,CS,E,MS

Utah
Hewlett-Packard Co.
3530 W. 2100 South
SALT LAKE CITY, UT 84119
Tel: (801) 974-1700
A,CH,CS,E,MS

Virginia
Hewlett-Packard Co.
4305 Cox Road
GLEN ALLEN, VA 23060
P.O. Box 9669
RICHMOND, VA 23228
Tel: (804) 747-7750
A,CH,CS,E,MS

Washington
Hewlett-Packard Co.
15815 S.E. 37th Street
BELLEVUE, WA 98006
Tel: (206) 643-4000
A,CH,CM,CS,E,MP

Hewlett-Packard Co.
Suite A
708 North Argonne Road
SPOKANE, WA 99212
Tel: (509) 922-7000
CH,CS

West Virginia
Hewlett-Packard Co.
4604 MacCorkle Ave.
P.O. Box 4297
CHARLESTON, WV 25304
Tel: (304) 925-0492
A,MS

Wisconsin
Hewlett-Packard Co.
150 S. Sunny Slope Road
BROOKFIELD, WI 53005
Tel: (414) 784-8800
A,CH,CS,E*,MP

URUGUAY

Pablo Ferrando S.A.C. e I.
Avenida Italia 2877
Casilla de Correo 370
MONTEVIDEO
Tel: 80-2586
Telex: Public Booth 901
A,CM,E,M

VENEZUELA

Hewlett-Packard de Venezuela C.A.
3RA Transversal Los Ruices Norte
Edificio Segre 1, 2 & 3
Apartado 50933
CARACAS 1071
Tel: 239-4133
Telex: 251046 HEWPACK
A,CH,CS,E,MS,P

Hewlett-Packard de Venezuela C.A.
Calle-72-Entre 3H y 3Y, No. 3H-40
Edificio Ada-Evelyn, Local B
Apartado 2646
4001, **MARACAIBO**, Estado Zulia
Tel: (061) 80.304
C,E*

Hewlett-Packard de Venezuela C.A.
Calle Vargas Rondon
Edificio Seguros Carabobo, Piso 10
VALENCIA
Tel: (041) 51 385
CH,CS,P

Bioelectronica Medica C.A.
Calle Buen Pastor
Edif. Cota Mil-Piso 2 y Semi Sotano 1
Boleita Norte
Apartado 50710 ***CARACAS*** *1050A*
Tel: 239 84 41
Telex: 26518

ZIMBABWE

Field Technical Sales
45 Kelvin Road, North
P.B. 3458
SALISBURY
Tel: 705 231
Telex: 4-122 RH
C,E,M,P

July 1983 5952-6900

Indicates main office

HP distributors are printed in italics.

Part No. 14752-90001

PRINTED IN U.S.A.

HP Binder Part No. 9282-0972

Printed in U.S.A.